LLM-Based Autonomous Multi-Agent Systems: A Comprehensive Survey

Anonymous Authors
Department of Computer Science
University Name
anonymous@acl2027.org

Abstract

The emergence of Large Language Models (LLMs) as cognitive cores for autonomous agents has catalyzed a paradigm shift from single-agent systems to sophisticated multi-agent architectures. This survey provides a comprehensive examination of LLM-based Multi-Agent Systems (LLM-MAS), systematically analyzing research developments from 2022 to 2026. We present a unified taxonomy covering four fundamental dimensions: (1) **Architecture Paradigms**—encompassing organizational patterns, control structures, and agent specialization; (2) **Coordination Mechanisms**—including communication protocols (MCP, ACP, A2A, ANP), coordination strategies, and consensus algorithms; (3) **Planning Strategies**—spanning task decomposition, hierarchical planning, and external module integration; and (4) **Evaluation Metrics**—addressing task performance, coordination quality, and system properties. Through systematic analysis of over 100 papers, we identify key research themes, map application domains from code generation to scientific discovery, and articulate open challenges including scalability to 100+ agents, protocol standardization, and multi-agent alignment. This survey serves as both a technical reference and a roadmap for future research in LLM-based multi-agent systems.

1 Introduction

1.1 Motivation and Background

The emergence of Large Language Models (LLMs) has fundamentally transformed the landscape of artificial intelligence, enabling machines to understand, reason, and communicate with unprecedented capability. What began with the breakthrough of transformer architectures has evolved into sophisticated systems capable of complex reasoning through techniques such as Chain-of-Thought prompting Wei et al. [2022], retrieval-augmented generation, and instruction following. This evolution has catalyzed a paradigm shift: LLMs are no longer merely passive text generators but are increasingly being deployed as the cognitive cores of *autonomous agents*—systems that can perceive their environment, reason about goals, and execute actions to achieve specified objectives.

The trajectory from single-agent to multi-agent systems represents a natural progression in this field. Early LLM-powered agents, exemplified by systems such as AutoGPT and BabyAGI, demonstrated the potential for autonomous task completion but also revealed fundamental limitations: monolithic agent architectures struggle with task complexity, exhibit bounded reasoning capabilities, and face challenges in scaling to diverse domains. These limitations have motivated researchers to explore *multi-agent architectures*, where specialized agents collaborate, coordinate, and collectively reason to solve problems beyond the reach of any single agent.

This transition mirrors historical developments in distributed systems and multi-agent reinforcement learning, where the decomposition of complex tasks across multiple specialized entities has consistently demonstrated superior performance, robustness, and scalability. However, LLM-based Multi-Agent Systems (LLM-MAS) introduce unique characteristics that distinguish them from their predecessors: the ability to communicate in natural language, the capacity for semantic reasoning, and the potential for emergent behaviors arising from the interplay of large-scale language models.

The rapid proliferation of LLM-MAS frameworks—from open-source libraries such as Agents Zhou et al. [2023] to specialized systems for code generation Shen et al. [2024], financial decision-making Yu et al.

[2024], and scientific discovery Rothfarb et al. [2025]—underscores both the timeliness and significance of this research direction. Yet, despite this explosive growth, the field lacks a unified theoretical framework and systematic taxonomy to guide researchers and practitioners through this complex landscape.

1.2 Problem Definition

The central challenge addressed by LLM-MAS research can be framed as follows: *How can multiple LLM-powered autonomous agents be organized, coordinated, and orchestrated to collectively solve complex tasks that exceed the capabilities of individual agents?*

This challenge encompasses several interrelated dimensions:

Architectural Design. What organizational structures—hierarchical, flat, or self-organizing—best support different task domains? How should agents be specialized, and what internal capabilities should each agent possess?

Coordination Mechanisms. How should agents communicate, share information, and reach consensus? What protocols and standards should govern inter-agent interactions?

Planning and Reasoning. How can complex tasks be decomposed and distributed across agents? What planning strategies enable effective multi-agent collaboration?

Evaluation. How should multi-agent systems be assessed? What metrics capture both individual agent performance and collective system behavior?

The absence of standardized definitions, taxonomies, and evaluation frameworks has led to fragmented research efforts, making it difficult to compare approaches, identify best practices, and chart future directions. This survey addresses these gaps by providing the first comprehensive examination of LLM-MAS as a unified research domain.

1.3 Formal Definition of LLM-MAS

We formally define an **LLM-based Multi-Agent System (LLM-MAS)** as follows:

Definition 1 (LLM-MAS)**.** *An LLM-based Multi-Agent System is a tuple $\mathcal{M} = \langle \mathcal{A}, \mathcal{E}, \mathcal{C}, \mathcal{P}, \mathcal{G} \rangle$ where:*

- *$\mathcal{A} = \{a_1, a_2, \ldots, a_n\}$ is a set of n autonomous agents, where each agent a_i is powered by one or more Large Language Models serving as its cognitive core;*
- *$\mathcal{E}$ is the shared environment in which agents operate, perceive states, and execute actions;*
- *$\mathcal{C} : \mathcal{A} \times \mathcal{A} \to \Sigma^*$ is the communication function defining message exchange between agents, where Σ^* denotes the set of natural language messages;*
- *$\mathcal{P} : \mathcal{T} \to \mathcal{A}^*$ is the planning function that decomposes tasks $t \in \mathcal{T}$ into agent assignments and action sequences;*
- *$\mathcal{G}$ is the global objective function that the multi-agent system aims to optimize through collective behavior.*

This definition distinguishes LLM-MAS from traditional multi-agent systems in several key aspects: (1) agents communicate primarily through natural language (Σ^*), enabling flexible and semantic-rich interactions; (2) the cognitive capabilities of each agent derive from LLMs, providing emergent reasoning abilities; and (3) the system exhibits both deliberate coordination through $\mathcal{C}$ and $\mathcal{P}$, as well as emergent collective behaviors.

1.4 Scope and Contributions

This survey provides a systematic examination of LLM-based Multi-Agent Systems, covering research developments from 2022 to 2026. We focus on systems where LLMs serve as the primary reasoning engine for autonomous agents engaged in collaborative task-solving. Our scope explicitly includes:

- **Architecture paradigms:** Organizational patterns, control structures, and agent specialization strategies;
- **Coordination mechanisms:** Communication protocols (MCP, ACP, A2A, ANP), coordination strategies, and consensus algorithms;
- **Planning strategies:** Task decomposition, hierarchical planning, and external module integration;
- **Evaluation methodologies:** Benchmark suites, metrics, and evaluation frameworks;
- **Applications:** Domain-specific implementations across code generation, finance, healthcare, scientific discovery, robotics, and gaming.

We explicitly exclude: (1) traditional multi-agent systems without LLM-based reasoning; (2) single-agent LLM systems; and (3) multi-agent systems where LLMs serve only peripheral roles (e.g., natural language interfaces without decision-making authority).

Key Contributions. This survey makes the following contributions to the field:

1. **Unified Taxonomy:** We present the first comprehensive taxonomy for LLM-MAS, organizing the field along four fundamental dimensions—Architecture, Coordination, Planning, and Evaluation—with fine-grained subcategories that enable systematic classification and comparison of approaches.

2. **Systematic Literature Analysis:** We analyze over 100 research papers from top-tier venues (NeurIPS, ICML, ICLR, ACL, EMNLP, AAAI, AAMAS) and arXiv, identifying key research themes, methodological patterns, and emerging trends.

3. **Formal Framework:** We provide formal definitions and theoretical grounding for LLM-MAS concepts, enabling rigorous analysis and comparison of different approaches.

4. **Application Mapping:** We systematically map LLM-MAS applications across diverse domains, identifying domain-specific architectural patterns and coordination requirements.

5. **Research Roadmap:** We identify critical open challenges—including scalability to 100+ agents, protocol standardization, multi-agent alignment, and evaluation standardization—and propose concrete directions for future research.

1.5 Significance and Timeliness

The study of LLM-MAS is both timely and significant for several reasons:

Exponential Growth. The field has experienced exponential growth, with publications increasing from a handful in 2022 to over 40 papers in 2025 alone. This rapid expansion reflects both the practical demand for multi-agent solutions and the theoretical richness of the research questions.

Industry Adoption. Major technology companies have invested heavily in LLM-MAS infrastructure, with frameworks such as Microsoft's AutoGen, LangChain's multi-agent capabilities, and emerging protocol standards (MCP from Anthropic, A2A from Google) indicating industrial significance.

Cross-Disciplinary Impact. LLM-MAS research draws from and contributes to multiple disciplines: natural language processing, multi-agent systems, distributed computing, human-computer interaction, and domain-specific applications (healthcare, finance, robotics).

Foundational Questions. Research in this area addresses fundamental questions about artificial intelligence: How can multiple intelligent systems collaborate effectively? What communication protocols enable robust coordination? How can we ensure safety and alignment in multi-agent settings?

The convergence of these factors makes LLM-MAS a critical area for systematic survey and analysis. By establishing a unified framework for understanding this rapidly evolving field, this survey aims to accelerate research progress and facilitate the development of more capable, reliable, and beneficial multi-agent systems.

1.6 Paper Organization

The remainder of this survey is organized as follows:

Section 2 provides background and preliminaries, tracing the evolution from single-agent to multi-agent systems and establishing foundational concepts.

Section 3 presents our taxonomy of architecture paradigms, examining organizational patterns, control structures, and agent specialization strategies.

Section 4 analyzes coordination mechanisms, including communication protocols, coordination strategies, and consensus algorithms.

Section 5 examines planning strategies, covering task decomposition, plan generation, and the integration of external modules.

Section 6 discusses evaluation methodologies, presenting benchmark suites, metrics, and evaluation frameworks specific to multi-agent settings.

Section 7 surveys applications across diverse domains, identifying domain-specific patterns and requirements.

Section 8 articulates open challenges and future research directions.

Section 9 concludes with a summary of contributions and an outlook on the field's trajectory.

Throughout this survey, we provide comparative tables, architectural diagrams, and structured analyses to facilitate understanding and enable researchers to quickly identify relevant prior work and promising research directions.

2 Background and Preliminaries

2.1 From Single-Agent to Multi-Agent Systems

The evolution from single-agent to multi-agent LLM systems represents a natural progression in AI architecture. Early LLM-powered agents, exemplified by systems such as AutoGPT and BabyAGI, demonstrated the potential for autonomous task completion but also revealed fundamental limitations: monolithic agent architectures struggle with task complexity, exhibit bounded reasoning capabilities, and face challenges in scaling to diverse domains.

Chain-of-Thought prompting Wei et al. [2022] established that LLMs could perform complex reasoning through intermediate steps. ReAct Yao et al. [2022] extended this by interleaving reasoning with action, enabling agents to interact with external environments. These foundational works paved the way for multi-agent systems where specialized agents collaborate, coordinate, and collectively reason to solve problems beyond the reach of any single agent.

2.2 Foundational Concepts

Agent Definition. An autonomous agent is a system that can perceive its environment, reason about goals, and execute actions to achieve specified objectives. LLM-based agents use large language models as their cognitive core, enabling natural language understanding, generation, and reasoning.

Multi-Agent System (MAS). A multi-agent system consists of multiple autonomous agents that interact within a shared environment. Key characteristics include:

- **Autonomy**: Agents operate without direct human intervention
- **Local Views**: No agent has full global knowledge
- **Decentralization**: No single designated controlling agent

LLM-MAS Distinction. LLM-based multi-agent systems differ from traditional MAS in several key aspects: (1) agents communicate primarily through natural language; (2) the cognitive capabilities derive from LLMs, providing emergent reasoning abilities; and (3) the system exhibits both deliberate coordination and emergent collective behaviors.

2.3 Related Surveys and Positioning

Several surveys have examined aspects of LLM agents. Huang et al. [2024b] survey planning in LLM agents, providing a taxonomy of planning methods. Mohammadi et al. [2025] propose a framework for evaluating LLM agents. Zhu et al. [2024] provide a comprehensive taxonomy for multi-agent deep reinforcement learning with communication. This survey differs by providing the first unified examination of LLM-based multi-agent systems as a distinct research paradigm, integrating insights across architecture, coordination, planning, and evaluation.

3 Architectural Paradigms

The architecture of an LLM-based Multi-Agent System (LLM-MAS) fundamentally determines how agents are organized, how they interact, and how computational resources are allocated. Unlike traditional multi-agent systems where agents are typically programmed with fixed behaviors, LLM-MAS architectures must accommodate the unique characteristics of large language models: their natural language interfaces, emergent reasoning capabilities, and substantial computational requirements. This section presents a comprehensive taxonomy of LLM-MAS architectures, covering the spectrum from single-LLM multi-agent configurations to fully decentralized multi-LLM systems, and from rigid hierarchical structures to adaptive self-organizing frameworks.

3.1 Taxonomy Overview

We propose a three-dimensional taxonomy for classifying LLM-MAS architectures. The three primary dimensions are:

1. **Organization Pattern**: The structural arrangement of agents, ranging from flat peer-to-peer networks to deeply hierarchical trees.
2. **Specialization Level**: The degree to which agents differ in their capabilities, roles, and underlying models.
3. **Dynamicity**: The extent to which the architecture can adapt its structure during runtime.

These dimensions are largely orthogonal, allowing for a rich design space. For instance, a system might employ a hierarchical organization with heterogeneous specialists in a static configuration, or a flat organization with homogeneous agents that dynamically reconfigure based on task demands.

3.2 Single-LLM Multi-Agent Architectures

In single-LLM multi-agent architectures, a single language model serves as the cognitive engine for multiple agent roles. This paradigm leverages the role-playing capabilities of LLMs, where different prompts and system messages induce different "personalities" or functional roles within the same underlying model. This approach offers significant advantages in terms of deployment simplicity and cost efficiency, as it requires maintaining only one model instance.

3.2.1 Role-Based Agent Instantiation

The foundational work on role-based multi-agent systems was established by Li et al. [2023], who introduced CAMEL (Communicative Agents for "Mind" Exploration of Large Language Model Society). CAMEL demonstrates that a single LLM can simulate multiple agents through carefully designed "inception prompting," where each agent role is defined through a combination of task descriptions and behavioral constraints. The framework enables autonomous cooperation between agents without human intervention, using role-playing to maintain conversation coherence.

Building on this foundation, Qian et al. [2024] developed ChatDev, a software development framework where a single LLM instantiates multiple specialized roles including programmers, reviewers, and testers. ChatDev introduces the concept of "chat chains" that guide what agents communicate about, and "communicative dehallucination" mechanisms that reduce hallucinations through structured dialogue. The framework demonstrates that role-based single-LLM systems can effectively simulate complex collaborative workflows, achieving notable success in automated software development tasks.

3.2.2 Advantages and Limitations

Single-LLM architectures offer several compelling advantages:

- **Deployment Simplicity**: Only one model instance needs to be hosted, reducing infrastructure complexity and cost.
- **Consistent Knowledge Base**: All agents share the same underlying knowledge, reducing inconsistencies in factual understanding.
- **Simplified Communication**: Agents communicate through natural language in a shared representational space.

However, these architectures also face significant limitations:

- **Computational Bottleneck**: Sequential processing of multiple agent turns can create latency issues, particularly for complex multi-step tasks.
- **Limited Specialization**: While role-playing can induce behavioral differences, all agents share the same fundamental capabilities and limitations of the underlying model.
- **Context Window Constraints**: Long multi-agent conversations can exceed context windows, requiring careful management of conversation history.

3.2.3 Representative Frameworks

CAMEL [Li et al., 2023] pioneered the communicative agent paradigm, demonstrating that LLMs can autonomously engage in role-playing conversations to solve complex tasks. The framework uses inception prompting to establish agent roles and maintains conversation coherence through structured dialogue patterns.

ChatDev [Qian et al., 2024] applies single-LLM multi-agent principles to software development, creating a virtual software company where agents assume roles such as CEO, CTO, programmers, and testers. The framework's chat chain structure guides agents through phases including coding, code review, and testing.

AgentVerse [Chen et al., 2023] provides a general framework for single-LLM multi-agent collaboration, emphasizing the emergence of social behaviors among agents. The framework demonstrates that groups of agents can outperform single agents through effective collaboration, while also identifying potential negative emergent behaviors that must be managed.

3.3 Multi-LLM Multi-Agent Architectures

Multi-LLM architectures employ multiple language model instances, potentially with different capabilities, sizes, or specializations. This paradigm enables true heterogeneity among agents, where different models can bring distinct strengths to collaborative tasks.

3.3.1 Heterogeneous Model Composition

The motivation for multi-LLM architectures stems from the recognition that different language models have different strengths. A large, capable model might excel at complex reasoning but be computationally expensive, while a smaller model might be faster and cheaper but less capable. Multi-LLM architectures can combine these complementary strengths.

Shen et al. [2024] explicitly address this trade-off in their Multi-LLM Agent framework, which decomposes agent functionality into three components: a planner for high-level reasoning, a caller for tool invocation, and a summarizer for output generation. Their work demonstrates that small LLMs are weak tool learners, but when combined with larger models in a multi-LLM architecture, the overall system can achieve strong performance while maintaining cost efficiency.

3.3.2 Model Specialization Strategies

Multi-LLM systems can employ various specialization strategies:

Functional Specialization Different models are assigned to different functional roles based on their capabilities. For example, a code-specialized model might handle programming tasks while a general-purpose model manages coordination and communication.

Domain Specialization Models fine-tuned on specific domains (e.g., medical, legal, financial) are deployed as domain experts within a larger multi-agent system. Yang et al. [2024a] employ this approach in MedAide, where specialized medical agents collaborate to address complex healthcare queries.

Size-Based Hierarchy Larger, more capable models serve as coordinators or managers, while smaller, faster models handle routine tasks. Yu et al. [2024] implement this pattern in FinCon, where a manager agent (using a larger model) coordinates analyst agents (using smaller models) for financial decision-making.

3.3.3 Cross-Model Communication

A key challenge in multi-LLM architectures is ensuring effective communication between agents powered by different models. Unlike single-LLM systems where agents share a common representational space, multi-LLM systems must bridge potential differences in:

- **Output Formats**: Different models may have different output styles and formatting conventions.
- **Reasoning Patterns**: Models may approach problems differently, requiring translation or alignment.
- **Knowledge Bases**: Different models may have different knowledge cutoffs or training data.

Recent work on agent communication protocols, discussed in Section 4, addresses these challenges through standardized message formats and semantic alignment mechanisms.

3.3.4 Representative Frameworks

AutoGen [?] provides a flexible framework for building multi-agent conversations, supporting both single-LLM and multi-LLM configurations. AutoGen agents are customizable and conversable, capable of operating in various modes that combine LLM inference, human input, and tool use. The framework's key innovation is its flexible approach to defining agent interaction behaviors through both natural language and code.

MetaGPT [?] introduces the concept of meta-programming for multi-agent collaboration, encoding Standardized Operating Procedures (SOPs) into prompt sequences. The framework uses an assembly line paradigm where different agents assume specialized roles (product manager, architect, engineer, etc.) in a software development pipeline. MetaGPT demonstrates that structured workflows can significantly reduce the cascading hallucinations that plague naive multi-agent chaining.

S-Agents [Chen et al., 2024] proposes a tree-of-agents structure with an "hourglass" architecture that balances horizontal collaboration among peers with vertical hierarchical coordination. The framework specifically addresses open-ended environments where agents must adapt to evolving task requirements.

3.4 Hierarchical vs. Flat Architectures

The organization pattern dimension distinguishes between hierarchical and flat architectures, each with distinct characteristics, advantages, and appropriate use cases.

3.4.1 Flat (Peer-to-Peer) Architectures

In flat architectures, agents operate as peers without designated leaders or coordinators. Communication flows directly between agents, and decisions emerge from collective deliberation rather than top-down direction.

Multi-Agent Debate A prominent pattern in flat architectures is multi-agent debate, where agents engage in dialectic reasoning to arrive at conclusions. **?** demonstrated that debate between multiple LLM instances can improve reasoning accuracy, as agents challenge each other's assumptions and correct errors. This approach has been extended to various domains including fact-checking [Theologitis and Suciu, 2025] and security analysis [Li et al., 2025c].

Consensus Mechanisms Flat architectures require mechanisms for reaching consensus when agents disagree. Common approaches include:

- **Voting**: Agents vote on proposals, with majority or weighted voting determining outcomes.
- **Iterative Refinement**: Agents take turns refining a shared solution until convergence.
- **Emergent Consensus**: Through sustained dialogue, agents naturally converge toward agreement.

Jo and Park [2025] extend these mechanisms to handle Byzantine failures, where some agents may provide incorrect or malicious responses. Their Byzantine-robust decentralized coordination framework demonstrates that flat architectures can achieve reliable consensus even in adversarial settings.

Advantages and Challenges Flat architectures offer high fault tolerance (no single point of failure), natural scalability (agents can be added without restructuring), and democratic decision-making. However, they face challenges in coordination efficiency (requiring more communication rounds), potential for deadlock (when agents cannot reach consensus), and difficulty in maintaining global coherence for complex tasks.

3.4.2 Hierarchical Architectures

Hierarchical architectures organize agents in tree-like structures with designated coordinators at various levels. This pattern mirrors human organizational structures and enables efficient task decomposition and allocation.

Manager-Worker Pattern The most common hierarchical pattern is the manager-worker configuration, where a manager agent decomposes tasks and allocates subtasks to worker agents. Xiao et al. [2024] implement this pattern in TradingAgents, where a chief analyst coordinates specialized analysts covering different market sectors. Similarly, Yu et al. [2024] employ a manager-analyst hierarchy in FinCon, demonstrating that hierarchical structures can effectively model real-world organizational workflows.

Multi-Level Hierarchies More complex tasks may require multiple levels of hierarchy. Chen et al. [2024] propose a tree-of-agents structure where intermediate nodes serve as both coordinators (for their subtrees) and workers (for their parent nodes). This "hourglass" architecture enables efficient information flow while maintaining the benefits of hierarchical decomposition.

Advantages and Challenges Hierarchical architectures offer efficient task decomposition, clear responsibility assignment, and scalable coordination (communication complexity grows logarithmically rather than quadratically). However, they introduce single points of failure (manager nodes), potential for misalignment between levels, and overhead in maintaining hierarchical structure.

3.4.3 Hybrid Architectures

Many practical systems employ hybrid architectures that combine hierarchical and flat elements. For example, a system might use hierarchical task decomposition at the top level while allowing peer-to-peer collaboration among workers at the bottom level. Zhao et al. [2024] demonstrate this approach in their hierarchical auto-organizing system, where agents dynamically form hierarchical structures for specific tasks while maintaining peer relationships for information sharing.

3.5 Centralized vs. Decentralized Control

Closely related to the hierarchical-flat distinction is the question of control distribution: whether decision-making authority is concentrated in a central coordinator or distributed among agents.

3.5.1 Centralized Coordination

In centralized architectures, a single agent or component serves as the coordination hub, managing task allocation, monitoring progress, and resolving conflicts.

Orchestrator Pattern The orchestrator pattern centralizes all coordination decisions in a single agent. Yu et al. [2024] implement this pattern with a "conceptual verbal reinforcement" mechanism where the central manager provides feedback to analysts. This enables consistent strategic direction but creates a bottleneck for complex tasks.

Blackboard Systems A variant of centralized coordination uses a shared "blackboard" or working memory that all agents can read from and write to. Pugachev [2025] implement a CRDT-based blackboard for multi-agent code generation, enabling concurrent contributions while maintaining consistency. This approach reduces coordination overhead while maintaining a shared state.

Limitations Centralized architectures face inherent scalability limitations, as the coordinator becomes a bottleneck as the number of agents grows. They also suffer from single points of failure, though redundancy mechanisms can mitigate this risk.

3.5.2 Decentralized Coordination

Decentralized architectures distribute coordination responsibilities among all agents, requiring explicit mechanisms for achieving coordination without central direction.

Distributed Consensus Decentralized systems must solve the consensus problem: how do distributed agents agree on decisions without a central authority? Jo and Park [2025] adapt classical distributed systems consensus protocols to LLM-MAS, demonstrating that agents can reach reliable consensus even when some agents provide incorrect outputs.

Market-Based Coordination An alternative approach uses market mechanisms for coordination. Tasks are "auctioned" to agents, who bid based on their capabilities and current workload. This approach, inspired by computational economies, enables efficient resource allocation without central planning.

Emergent Coordination Some systems rely on emergent coordination, where coherent behavior arises from simple local rules rather than explicit coordination mechanisms. Chen et al. [2023] observe emergent social behaviors in their multi-agent framework, including both beneficial behaviors (collaboration, knowledge sharing) and detrimental ones (groupthink, cascade errors).

3.5.3 Hybrid Control Models

Practical systems often employ hybrid control models that balance centralized efficiency with decentralized robustness. Common patterns include:

- **Hierarchical with Local Autonomy**: Central coordinators handle high-level decisions while individual agents have autonomy within their domains.
- **Federated Coordination**: Multiple coordinators manage subgroups of agents, with peer-to-peer coordination among coordinators.
- **Adaptive Control**: The system dynamically shifts between centralized and decentralized modes based on task requirements and system state.

3.6 Static vs. Dynamic Architectures

The dynamicity dimension addresses whether the system architecture is fixed at design time or can adapt during runtime.

3.6.1 Static Architectures

In static architectures, the number of agents, their roles, and their relationships are predetermined and remain fixed throughout execution. This approach offers predictability and simplifies analysis but may be suboptimal for tasks with varying complexity or requirements.

Most current LLM-MAS frameworks employ static architectures. For example, ChatDev always instantiates the same set of software development roles, and MetaGPT follows a fixed assembly line structure. This predictability enables systematic optimization and debugging but limits adaptability.

3.6.2 Dynamic and Self-Organizing Architectures

Dynamic architectures can modify their structure during runtime, adding or removing agents, changing roles, or reorganizing relationships based on task demands or performance feedback.

Self-Organizing Systems Zhao et al. [2024] propose a hierarchical auto-organizing system where agents dynamically form organizational structures based on task requirements. The system can spawn new agents, merge existing ones, and reconfigure hierarchies without external intervention.

Population-Based Approaches Some systems maintain a population of agents and dynamically select subsets for specific tasks. Huynh et al. [2024] apply this approach to game playing, where a population of agents evolves through self-play and the most effective agents are selected for deployment.

Challenges Dynamic architectures face significant challenges:

- **Stability**: Rapid reorganization can lead to unstable behavior.
- **Coherence**: Maintaining coherent system behavior during structural changes is difficult.
- **Overhead**: The mechanisms for dynamic reorganization consume computational resources.

3.7 Agent Internal Architecture

Beyond system-level architecture, individual agents in LLM-MAS have internal architectures that determine their capabilities. Following the taxonomy proposed by Zhu et al. [2024], we identify five core modules:

1. **Perception Module**: Handles input from the environment, including multi-modal inputs (text, images, audio). Systems like Mobile-Agent [Wang et al., 2024] demonstrate sophisticated perception capabilities for mobile device interaction.

2. **Memory System**: Manages information persistence across interactions. Memory systems range from simple context windows to sophisticated architectures with episodic, semantic, and procedural memory components [Shkolnikov, 2026, Mody et al., 2026].

3. **Reasoning Module**: Implements deliberative processes including chain-of-thought reasoning, tree-of-thought exploration, and ReAct-style interleaved reasoning and acting [Huang et al., 2024b].

4. **Planning Module**: Handles task decomposition, plan generation, and replanning. Advanced planning modules integrate with external planning formalisms such as PDDL [Gestrin et al., 2024, Chu et al., 2025].

5. **Action Module**: Executes decisions through tool invocation, code generation, or environment interaction. Tool use capabilities are increasingly important, with frameworks like ToolScope [Liu et al., 2025] enhancing LLM tool selection and execution.

The sophistication of these internal modules varies significantly across frameworks. Simple agents might have only perception and action modules (directly mapping inputs to outputs), while advanced agents incorporate all five modules with sophisticated inter-module communication.

3.8 Comparative Analysis

Table 1 presents a comparative analysis of architectural patterns across key dimensions. The choice of architecture depends on task characteristics, scale requirements, and resource constraints.

Task Complexity For simple, well-defined tasks, flat architectures with homogeneous agents often suffice. Complex tasks requiring diverse expertise benefit from hierarchical structures with heterogeneous specialists.

Scale Requirements Small-scale systems (fewer than 10 agents) can use any architecture effectively. Large-scale systems (hundreds of agents) require hierarchical or decentralized architectures to manage coordination complexity.

Resource Constraints When computational resources are limited, single-LLM architectures with careful prompt engineering may be preferable. When resources permit, multi-LLM architectures can achieve better performance through specialization.

Robustness Requirements Mission-critical applications requiring high fault tolerance should prefer decentralized or self-organizing architectures that can tolerate individual agent failures.

Table 1: Comparison of Architectural Patterns in LLM-Based Multi-Agent Systems

Pattern	**Scalability**	**Coordination**	**Fault Tolerance**	**Complexity**	**Example Systems**
Flat/P2P	High	Decentralized	High	Low	CAMEL, Debate systems
Manager-Worker	Medium	Centralized	Low	Medium	TradingAgents, FinCon
Tree-of-Agents	Medium-High	Hierarchical	Medium	Medium	S-Agents
Modular Pipeline	Medium	Sequential	Low	Low	Planner-Caller-Summarizer
Self-Organizing	High	Adaptive	High	High	Hierarchical Auto-Org

3.9 Design Considerations and Trade-offs

Selecting an appropriate architecture involves navigating several fundamental trade-offs:

3.9.1 Efficiency vs. Robustness

Centralized architectures are typically more efficient (fewer coordination messages, faster decision-making) but less robust (single points of failure). Decentralized architectures sacrifice efficiency for robustness. The appropriate balance depends on application requirements: real-time systems may prioritize efficiency, while safety-critical systems may prioritize robustness.

3.9.2 Specialization vs. Flexibility

Highly specialized agents can achieve superior performance on their designated tasks but may struggle with unexpected situations. Generalist agents are more flexible but may not match specialist performance. Multi-LLM architectures can achieve both through strategic composition of specialist and generalist models.

3.9.3 Structure vs. Emergence

Highly structured architectures (fixed roles, predetermined workflows) provide predictability and debuggability but may miss opportunities for emergent solutions. Loosely structured architectures allow for emergent behaviors but may produce unpredictable or undesirable outcomes. MetaGPT's approach of encoding SOPs represents a middle ground, providing structure while preserving some flexibility.

3.9.4 Cost vs. Capability

Larger models and more agents increase capability but also cost. The relationship is not linear: a well-designed multi-agent system with smaller models can sometimes outperform a single large model while being more cost-effective. Shen et al. [2024] provide empirical evidence for this, showing that their multi-LLM architecture with task decomposition achieves strong performance with reduced computational costs.

3.10 Future Directions

The architecture of LLM-MAS continues to evolve rapidly. Several promising directions emerge from current research:

Adaptive Architectures Future systems may dynamically adjust their architecture based on task characteristics, performance feedback, and resource availability. This requires meta-level reasoning about system structure and the ability to reconfigure at runtime.

Neural-Symbolic Integration Combining the flexibility of neural LLMs with the rigor of symbolic reasoning systems could enable more reliable and interpretable architectures. Work on LLM+MAP [Chu et al., 2025] demonstrates this integration for robot task planning.

Human-Agent Teaming Architectures that effectively integrate human and AI agents remain an open challenge. This requires not only technical mechanisms for human-agent communication but also careful consideration of authority, accountability, and trust.

Scalable Coordination As LLM-MAS scale to hundreds or thousands of agents, new coordination mechanisms will be required. Insights from traditional distributed systems and multi-agent reinforcement learning [Zhu et al., 2024] will be valuable, but must be adapted to the unique characteristics of LLM-based agents.

In summary, the architectural design of LLM-MAS involves complex trade-offs across multiple dimensions. No single architecture is optimal for all tasks; rather, architects must carefully match architectural choices to application requirements, resource constraints, and desired system properties. The frameworks and patterns discussed in this section provide a foundation for making these design decisions systematically.

4 Coordination Mechanisms and Communication Protocols

Effective coordination among multiple agents is fundamental to the success of LLM-based multi-agent systems (LLM-MAS). Unlike single-agent systems where one LLM handles all aspects of a task, multi-agent systems must establish protocols and mechanisms for agents to communicate, negotiate, reach consensus, and organize themselves into effective teams. This section provides a comprehensive analysis of coordination mechanisms in LLM-MAS, covering agent communication languages, message passing protocols, negotiation strategies, consensus mechanisms, role assignment, and team formation. We discuss both explicit coordination mechanisms, where agents deliberately exchange information, and implicit coordination, where agents coordinate through shared environment observations or learned behaviors.

4.1 Agent Communication Languages

Agent communication languages (ACLs) provide the foundational vocabulary and semantics for inter-agent message exchange. The evolution from traditional ACLs to LLM-native protocols reflects a fundamental shift in how autonomous agents communicate.

4.1.1 Traditional Agent Communication Languages

Historically, multi-agent systems relied on standardized ACLs such as the Knowledge Query Manipulation Language (KQML) Labrou and Finin [1997] and the Foundation for Intelligent Physical Agents (FIPA) ACL FIPA [2002]. These languages defined:

- **Performatives**: Speech act-based primitives (e.g., *inform*, *request*, *query*) that specify the communicative intent of messages.
- **Content Languages**: Formal representations for message content, often based on first-order logic or description logics.
- **Interaction Protocols**: Standardized conversation patterns such as contract net, English auction, or request-respond sequences.

KQML, developed in the early 1990s, introduced the concept of performatives as communication primitives with associated semantics based on speech act theory Labrou and Finin [1997]. FIPA ACL extended this foundation with formal semantics based on social commitments, providing a more rigorous framework for verifying agent communication behaviors FIPA [2002].

4.1.2 LLM-Native Communication Protocols

The emergence of LLM-based agents has prompted the development of new communication protocols that leverage natural language capabilities while providing structured semantics for reliable coordination. Recent surveys identify several emerging protocols designed specifically for LLM agent interoperability Ehtesham et al. [2025]:

The Model Context Protocol (MCP), introduced by Anthropic in 2024, focuses on enabling LLM agents to share context and tool capabilities across different implementations Ehtesham et al. [2025]. MCP provides a standardized interface for tools to expose their capabilities and for agents to invoke these tools with proper context management.

The Agent Network Protocol (ANP) proposes a three-layer architecture inspired by network protocol stacks Chang et al. [2025]:

1. **Application Layer**: Task semantics, intent specification, and contextual information.
2. **Transport Layer**: Message routing, delivery guarantees, and reliability mechanisms.
3. **Network Layer**: Agent discovery, addressing, and topology management.

Li et al. Li et al. [2025d] argue for urgent standardization of LLM Agent Communication Protocol (LACP), drawing inspiration from telecommunications protocols to provide reliable, scalable, and interoperable communication infrastructure. Their proposal emphasizes the need for standardized message formats, addressing schemes, and routing protocols to enable large-scale LLM agent networks.

4.1.3 Semantic Challenges in LLM Communication

A critical challenge in LLM-based agent communication is maintaining semantic consistency across message exchanges. Shekkizhar et al. Shekkizhar et al. [2025] identify an “echoing” phenomenon where LLM agents can lose their identity when communicating with each other, leading to convergence toward similar responses rather than maintaining diverse perspectives essential for effective collaboration. This highlights the need for protocols that preserve agent individuality while enabling productive coordination.

4.2 Message Passing Mechanisms

Message passing is the fundamental mechanism through which agents exchange information in LLM-MAS. The design of message passing systems significantly impacts system performance, scalability, and coordination effectiveness.

4.2.1 Communication Patterns

Research on multi-agent deep reinforcement learning with communication (Comm-MADRL) identifies several canonical communication patterns Zhu et al. [2024]:

Broadcast Communication: In broadcast patterns, each agent sends messages to all other agents. This pattern is suitable for global state synchronization and consensus building but scales poorly with the number of agents due to quadratic message complexity Zhu et al. [2024].

Targeted Communication: Point-to-point messaging enables efficient pipeline processing where agents pass information along a predefined path. This pattern is common in modular architectures such as the Planner-Caller-Summarizer framework proposed by Shen et al. Shen et al. [2024].

Hierarchical Communication: Tree-structured message passing, where a manager agent coordinates with worker agents, provides a balance between coordination efficiency and scalability. This pattern is employed in systems like TradingAgents Xiao et al. [2024] and FinCon Yu et al. [2024].

Ring Communication: Sequential message passing in a ring topology is effective for iterative reasoning and debate protocols, where each agent refines the output of the previous agent **?**.

4.2.2 Message Content and Structure

The content of inter-agent messages varies significantly across systems. Zhu et al. Zhu et al. [2024] propose a nine-dimensional taxonomy for analyzing communication content in multi-agent reinforcement learning:

1. **When to communicate**: Triggered by events, time intervals, or learned policies.
2. **What to communicate**: Raw observations, processed features, or abstract intentions.
3. **Who communicates**: All agents, selected agents, or learned communication graphs.
4. **Whom to address**: Broadcast, multicast, or unicast addressing.
5. **How to communicate**: Discrete symbols, continuous vectors, or natural language.
6. **Message aggregation**: Sum, attention-weighted, or graph neural network aggregation.
7. **Integration with learning**: End-to-end trainable or fixed protocols.
8. **Communication budget**: Fixed limits or adaptive allocation.
9. **Temporal aspects**: Synchronous or asynchronous messaging.

For LLM-based agents, natural language messages provide flexibility and interpretability but introduce challenges in parsing and semantic consistency. Peng et al. Peng et al. [2025] demonstrate that structured message formats with explicit verification mechanisms can improve collaboration under information asymmetry.

4.2.3 Bandwidth and Efficiency Considerations

Communication bandwidth is a critical constraint in multi-agent systems. Research on emergent communication demonstrates that low-bandwidth communication often emerges naturally in cooperative multi-agent learning **?**. For LLM agents, token efficiency becomes a practical concern, as each message consumes computational resources.

Cheng et al. Cheng et al. [2026] propose interference-aware K-step reachable communication (IA-KRC) that confines message passing to physically or logically accessible neighbors while optimizing partner selection to minimize interference. This approach demonstrates improved scalability in complex topological environments.

4.3 Negotiation Protocols

Negotiation protocols enable agents to reach agreements on resource allocation, task assignment, and joint action selection. In LLM-MAS, negotiation mechanisms range from simple bidding protocols to complex multi-round bargaining.

4.3.1 Contract Net Protocol

The Contract Net Protocol (CNP), originally proposed by Smith **?**, remains foundational for task allocation in multi-agent systems. In CNP:

1. A manager agent announces a task with specifications.
2. Contractor agents evaluate the task and submit bids.
3. The manager evaluates bids and awards the contract.
4. The selected contractor executes the task and reports results.

This protocol has been adapted for LLM-based systems where agents use natural language for task announcements and bids. The flexibility of LLMs enables more expressive task descriptions and bid justifications compared to traditional structured protocols.

4.3.2 Auction-Based Mechanisms

Auction-based negotiation provides a structured framework for competitive resource allocation. Common auction types in multi-agent systems include:

- **English Auction**: Open ascending bids until no higher bid is offered.
- **Vickrey Auction**: Sealed-bid auction where the highest bidder wins but pays the second-highest bid price.
- **Combinatorial Auction**: Agents bid on bundles of items, addressing interdependent resource allocation.

Marsa-Maestre et al. **?** demonstrate nonlinear negotiation approaches for complex network optimization, showing that simulated annealing-based negotiators can outperform particle swarm optimization in scenarios with complex interdependencies.

4.3.3 Bargaining and Multi-Round Negotiation

For scenarios requiring iterative refinement, multi-round bargaining protocols enable agents to propose and counter-propose solutions. Murukannaiah and Jonker **?** propose the Multiple Offers Protocol for Multilateral Negotiations with Partial Consensus (MOPaC), which allows subsets of agents to reach agreements without requiring full consensus among all parties.

In LLM-based systems, natural language enables more sophisticated bargaining strategies. Agents can justify their positions, propose compromises, and explain the rationale behind their proposals. However, this flexibility introduces challenges in ensuring negotiation convergence and preventing endless debate cycles.

4.4 Consensus Mechanisms

Consensus mechanisms enable multiple agents to reach agreement on decisions, beliefs, or actions. The choice of consensus mechanism significantly impacts system reliability, fault tolerance, and decision quality.

4.4.1 Voting-Based Consensus

Voting provides a straightforward mechanism for collective decision-making:

Majority Voting: The most common approach where the option receiving more than half of votes wins. Simple and efficient, but may not capture nuanced preferences.

Weighted Voting: Votes are weighted by agent expertise, reliability, or stake. Requires calibration of weights and may be sensitive to weight manipulation.

Ranked Choice Voting: Agents rank options rather than selecting a single choice. Captures more nuanced preferences but increases computational complexity.

Wu et al. **?** analyze voting mechanisms in the context of LLM agent debates, finding that majority pressure can suppress independent correction and that effective teams are those that can overturn incorrect consensus.

4.4.2 Byzantine Fault Tolerance

In distributed systems where some agents may fail or behave maliciously, Byzantine fault-tolerant consensus protocols ensure reliable agreement. Jo and Park Jo and Park [2025] propose Byzantine-robust decentralized coordination for LLM agents, addressing scenarios where some agents may produce incorrect or malicious outputs.

Key requirements for Byzantine-robust consensus include:

- Detection and isolation of faulty agents
- Redundant computation with cross-validation

- Threshold-based agreement mechanisms
- Reputation tracking for agent reliability

4.4.3 Conflict-Free Replicated Data Types (CRDTs)

For coordination in distributed environments, CRDTs provide a mathematical foundation for eventual consistency without requiring synchronous coordination. Pugachev Pugachev [2025] applies CRDTs to multi-agent LLM code generation, enabling lock-free concurrent code development where multiple agents can work independently while their contributions are automatically merged.

CRDT-based coordination is particularly suitable for:

- Offline-capable systems where agents may be temporarily disconnected
- High-latency environments where synchronous coordination is impractical
- Systems requiring strong eventual consistency guarantees

4.4.4 Multi-Agent Debate

Multi-agent debate has emerged as a distinctive consensus mechanism for LLM-based systems. In debate protocols, agents present and defend positions, respond to critiques, and iteratively refine their arguments.

Research on multi-agent debate reveals nuanced findings about its effectiveness. Wu et al. **?** conduct controlled studies showing that intrinsic reasoning strength and group diversity are dominant drivers of debate success, while structural parameters such as order or confidence visibility offer limited gains. Their analysis identifies key behavioral patterns:

- Majority pressure can suppress independent correction
- Effective teams overturn incorrect consensus
- Rational, validity-aligned reasoning predicts improvement

TS-Debate **?** demonstrates modality-specialized collaborative debate for time series reasoning, where dedicated expert agents handle textual context, visual patterns, and numerical signals, coordinated through a structured debate protocol with reviewer agents.

4.5 Role Assignment and Specialization

Role assignment is fundamental to organizing multi-agent systems effectively. By assigning specialized roles to agents, systems can leverage expertise, reduce coordination overhead, and improve task performance.

4.5.1 Role Definition and Types

Roles in LLM-MAS can be categorized along several dimensions:

Functional Roles: Based on task-related capabilities

- *Planner*: Decomposes tasks and generates execution plans
- *Executor*: Carries out specific actions or tool invocations
- *Critic*: Evaluates outputs and provides feedback
- *Coordinator*: Manages inter-agent communication and task allocation

Domain Roles: Based on knowledge expertise

- *Domain Expert*: Specialized knowledge in specific fields
- *Generalist*: Broad but shallow knowledge across domains
- *Mediator*: Facilitates communication between specialists

Hierarchical Roles: Based on organizational position

- *Manager*: Makes high-level decisions and allocates resources
- *Worker*: Executes assigned subtasks
- *Observer*: Monitors system behavior and reports anomalies

4.5.2 Role Assignment Mechanisms

Role assignment can be performed through various mechanisms:

Static Assignment: Roles are predefined at system design time. This approach is simple but lacks flexibility for dynamic environments. Examples include the Planner-Caller-Summarizer architecture in Multi-LLM Agent Shen et al. [2024].

Dynamic Assignment: Roles are assigned at runtime based on task requirements and agent capabilities.

MedAide Yang et al. [2024a] employs dynamic role rotation where medical agents rotate through different specializations based on patient query types.

Emergent Assignment: Roles emerge from agent interactions and self-organization. Chen et al. Chen et al. [2024] demonstrate self-organizing agents that develop specialized roles through interaction in open-ended environments.

4.5.3 Role Specialization and Coordination

Effective role specialization requires balancing depth of expertise with coordination overhead. Highly specialized agents may excel at their designated tasks but struggle to coordinate with agents in other roles.

Yang et al. Yang et al. [2024a] propose a rotation agent collaboration mechanism that introduces dynamic role rotation and decision-level information fusion across specialized medical agents. This approach maintains specialization while ensuring cross-domain knowledge integration.

The Geometry of Dialogue framework **?** proposes an interaction-centric approach to automatic team composition. By constructing a "language model graph" that maps relationships between models from semantic coherence of pairwise conversations, the system identifies synergistic model clusters that reflect latent specializations, enabling automated formation of effective multi-agent teams.

4.6 Team Formation

Team formation addresses the challenge of selecting and organizing agents into effective collaborative structures. Unlike role assignment which focuses on individual agent functions, team formation considers the collective capabilities and interactions of agent groups.

4.6.1 Team Composition Strategies

Homogeneous Teams: All agents have identical capabilities and roles. Suitable for tasks requiring parallel processing or redundancy, but limited in handling diverse task requirements.

Heterogeneous Teams: Agents have diverse capabilities and specializations. Enables handling of complex, multi-faceted tasks but requires sophisticated coordination mechanisms.

Hierarchical Teams: Agents organized in a tree structure with managers and workers. Balances specialization with coordination efficiency.

Moslemi and Lee **?** investigate bilateral team formation in cooperative multi-agent reinforcement learning, finding that algorithmic properties in bilateral team formation significantly influence policy performance and generalization.

4.6.2 Team Formation Mechanisms

Centralized Formation: A coordinator agent selects team members based on task requirements and agent profiles. Used in systems like FinCon Yu et al. [2024] where a manager agent coordinates analyst agents.

Decentralized Formation: Agents self-organize into teams through local interactions and negotiation. Suitable for open, dynamic environments but may require more communication overhead.

Learning-Based Formation: Machine learning models predict optimal team compositions based on historical performance data. Furuya and Kitagawa **?** demonstrate that language model graphs can identify synergistic teams that outperform random baselines.

4.6.3 Dynamic Team Reconfiguration

In dynamic environments, teams must adapt their composition as task requirements change. Key challenges include:

- **Continuity**: Maintaining progress during team transitions
- **Knowledge Transfer**: Preserving relevant knowledge when agents join or leave
- **Coordination Overhead**: Minimizing disruption during reconfiguration

Chen et al. Chen et al. [2024] propose self-organizing agents that dynamically reconfigure team structure based on environmental demands, demonstrating effective adaptation in open-ended Minecraft environments.

4.7 Explicit vs. Implicit Coordination

Coordination mechanisms can be broadly categorized as explicit or implicit, each with distinct characteristics and applicability.

4.7.1 Explicit Coordination

Explicit coordination involves deliberate communication and negotiation among agents:

Characteristics:

- Structured message exchange with defined protocols
- Negotiable agreements and contracts
- Verifiable commitments and obligations
- Interpretable coordination processes

Advantages:

- Clear accountability and responsibility
- Auditability of coordination decisions
- Ability to enforce constraints and policies

Challenges:

- Communication overhead
- Latency in coordination decisions
- Dependency on protocol compliance

Explicit coordination is essential for applications requiring accountability, such as financial trading Xiao et al. [2024], Yu et al. [2024] and medical decision-making Yang et al. [2024a].

4.7.2 Implicit Coordination

Implicit coordination occurs without explicit message exchange, relying instead on shared context, conventions, or learned behaviors:

Mechanisms:

- **Shared Environment**: Agents observe and respond to environmental state changes caused by other agents
- **Conventions**: Pre-established patterns of behavior that enable coordination without communication
- **Emergent Coordination**: Learned behaviors that produce coordinated outcomes

Bredell et al. **?** demonstrate augmenting agent action spaces with conventions for improved cooperation in Hanabi, showing that implicit knowledge sharing through conventions can significantly enhance performance in partially observable settings with limited communication.

Advantages:

- Lower communication overhead
- Faster response times
- Robustness to communication failures

Challenges:

- Limited expressiveness
- Difficulty in handling novel situations
- Challenges in verification and debugging

4.7.3 Hybrid Approaches

Many practical systems combine explicit and implicit coordination. For example, in game-playing agents, explicit communication may be used for strategic planning while implicit coordination handles tactical execution Light et al. [2023].

Peters et al. Peters et al. [2025] provide a comprehensive survey of emergent language in multi-agent systems, identifying research gaps in bridging learned communication protocols with interpretable, human-aligned coordination mechanisms.

4.8 Coordination Quality Metrics

Evaluating coordination effectiveness requires metrics that capture both process quality and outcome quality.

4.8.1 Communication Efficiency

- **Message Count**: Total number of messages exchanged per task
- **Communication Rate**: Messages per unit time or per task completion
- **Information Density**: Useful information per message
- **Redundancy Ratio**: Duplicate or unnecessary information

4.8.2 Consensus Quality

- **Agreement Rate**: Proportion of decisions reaching consensus
- **Convergence Time**: Time or iterations to reach consensus
- **Consensus Correctness**: Accuracy of consensus decisions
- **Stability**: Persistence of consensus over time

4.8.3 Coordination Overhead

- **Time Overhead**: Time spent on coordination activities
- **Resource Overhead**: Computational resources for coordination
- **Opportunity Cost**: Task performance impact of coordination

Wu et al. Wu and Fard [2025] propose HumanEval-Comm for benchmarking communication competence in code generation, evaluating both correctness and communication quality.

4.9 Challenges and Open Problems

Despite significant progress, several challenges remain in coordination for LLM-MAS:

4.9.1 Scalability

Most existing coordination mechanisms scale poorly beyond tens of agents. The quadratic complexity of all-to-all communication and the overhead of consensus protocols limit applicability to large-scale systems. Hierarchical coordination and selective communication offer partial solutions but introduce their own trade-offs.

4.9.2 Heterogeneity

Coordinating agents with different capabilities, knowledge bases, and operational constraints requires flexible protocols that can accommodate diversity while maintaining coherence. Standardized communication protocols like MCP, ACP, A2A, and ANP aim to address interoperability but adoption remains limited.

4.9.3 Robustness

Coordination mechanisms must be robust to agent failures, communication disruptions, and adversarial manipulation. Byzantine fault-tolerant protocols provide theoretical foundations but practical implementation in LLM-based systems remains challenging.

4.9.4 Interpretability

Understanding why coordination succeeded or failed is essential for debugging and improvement. The opacity of LLM decision-making complicates attribution of coordination outcomes to specific agent behaviors or protocol features.

4.9.5 Human-Agent Coordination

As LLM-MAS increasingly operate alongside humans, coordination mechanisms must support human-agent collaboration. This requires protocols that are interpretable to humans, can accommodate human intervention, and respect human authority constraints.

4.10 Summary

Coordination mechanisms are fundamental to the effectiveness of LLM-based multi-agent systems. This section has examined the landscape of coordination approaches, from traditional agent communication languages to emerging LLM-native protocols, from explicit negotiation to implicit coordination through conventions and shared environment.

Key findings include:

- Communication protocols are evolving from structured ACLs to LLM-native protocols that leverage natural language while maintaining semantic consistency.
- Message passing patterns significantly impact system scalability and should be selected based on coordination requirements.
- Negotiation and consensus mechanisms range from simple voting to complex debate protocols, with trade-offs between efficiency and decision quality.
- Role assignment and team formation are critical for organizing effective multi-agent systems, with approaches ranging from static to dynamic and emergent.
- Both explicit and implicit coordination have roles to play, with hybrid approaches often providing the best balance.

The coordination mechanisms discussed in this section provide the foundation for the planning strategies examined in Section 5, where coordinated agents must decompose tasks, generate plans, and execute actions in pursuit of collective goals.

5 Planning Strategies and Task Decomposition

Planning constitutes a fundamental capability of LLM-based multi-agent systems, enabling agents to decompose complex tasks into executable actions, coordinate their behaviors, and adapt to dynamic environments. This section provides a comprehensive analysis of planning strategies in LLM-MAS, covering task decomposition techniques, planning algorithms, execution monitoring, error recovery mechanisms, and coordination protocols for both single-agent and multi-agent scenarios.

5.1 Overview of Planning in LLM Agents

Planning in LLM-based agents refers to the process of generating, selecting, and executing action sequences to achieve specified goals. Unlike traditional automated planning systems that rely on formal representations such as PDDL (Planning Domain Definition Language), LLM-based planning leverages the emergent reasoning capabilities of large language models to generate plans from natural language descriptions Huang et al. [2024b].

Huang et al. [2024b] propose a comprehensive taxonomy for understanding LLM agent planning, identifying five key dimensions: (1) task decomposition, (2) plan selection, (3) external module integration, (4) reflection, and (5) memory utilization. This framework highlights the multifaceted nature of planning in LLM agents, extending beyond simple action generation to encompass the full planning-execution-reflection lifecycle.

The integration of LLMs with traditional planning formalisms represents a promising research direction. Chu et al. [2025] introduce LLM+MAP, a framework that combines LLM reasoning with PDDL-based multi-agent planning for bimanual robot task planning. Their approach demonstrates that structured symbolic planning can complement LLM reasoning, achieving superior performance compared to pure LLM-based approaches on long-horizon manipulation tasks.

5.2 Task Decomposition Techniques

Task decomposition forms the foundation of planning in complex multi-agent systems, enabling the breakdown of high-level goals into manageable subtasks that can be distributed among agents. We identify four primary decomposition paradigms in LLM-MAS.

5.2.1 Goal-Based Decomposition

Goal-based decomposition recursively breaks high-level objectives into sub-goals until reaching atomic actions. This approach is intuitive and aligns naturally with hierarchical organizational structures. Hu et al. [2024] introduce AgentGen, a framework that enhances LLM planning abilities through automated environment and task generation. Their bidirectional evolution method (Bi-Evol) synthesizes planning tasks with smooth difficulty curves, enabling progressive training of decomposition capabilities.

The AgentGen framework demonstrates that instruction tuning with diverse planning trajectories significantly improves decomposition quality. Experiments on AgentBoard show that AgentGen-tuned Llama-3.1-8B surpasses GPT-3.5 in overall planning performance, while the 70B variant achieves state-of-the-art results on planning benchmarks.

5.2.2 Hierarchical Task Network (HTN) Decomposition

Hierarchical Task Networks provide a structured approach to decomposition through method hierarchies that specify how abstract tasks expand into concrete action sequences. Pellier et al. [2023] propose HDDL 2.1, extending the hierarchical domain definition language to support temporal and numerical constraints essential for real-world applications.

The integration of LLMs with HTN planning presents unique opportunities. Puerta-Merino et al. [2025] present a roadmap for LLM-HTN integration, proposing a taxonomy of integration methods across the hierarchical planning lifecycle. Their benchmark reveals that while pure LLM planners achieve only 3% correct plans with no valid hierarchical decompositions, the combination of LLM reasoning with HTN structure holds significant promise for complex planning domains.

5.2.3 Role-Based Decomposition

Role-based decomposition assigns subtasks to specialized agents based on their capabilities and expertise. This approach is particularly effective in heterogeneous multi-agent systems where agents possess distinct skills. Shen et al. [2024] demonstrate this principle in their multi-LLM agent framework, decomposing tool-use capabilities into three specialized roles: a planner for task decomposition, a caller for tool invocation, and a summarizer for result synthesis.

Their two-stage training paradigm—comprehensive backbone fine-tuning followed by role-specific specialization—enables smaller LLMs to achieve competitive performance on tool-use benchmarks. This modular decomposition facilitates individual component updates and enables the use of smaller, specialized models for each capability.

5.2.4 Temporal Decomposition

Temporal decomposition structures tasks according to time-based constraints and dependencies, enabling parallel execution and efficient resource utilization. This approach is essential for domains requiring coordinated multi-agent actions with temporal ordering constraints.

Wu et al. [2024] introduce CATP-LLM for cost-aware tool planning, enabling LLMs to generate multi-branch non-sequential plans that optimize for execution cost. Their approach employs reinforcement learning to learn cost-aware planning strategies, demonstrating significant improvements in tool selection efficiency.

5.3 Planning Algorithms and Approaches

Planning algorithms in LLM-MAS span a spectrum from single-shot generation to iterative refinement and hierarchical reasoning. We examine the principal approaches and their characteristics.

5.3.1 Single-Shot Planning

Single-shot planning generates complete plans through a single LLM invocation, leveraging the model's parametric knowledge and in-context reasoning. While efficient, this approach may struggle with complex, long-horizon tasks requiring multi-step reasoning.

Huang et al. [2024a] investigate strategies for end-to-end LLM plan generation, proposing the Longest Contiguous Common Subsequence (LCCS) reward for reinforcement learning-based plan optimization. Their analysis reveals that progress-oriented rewards outperform perfection-seeking objectives, suggesting that incremental improvement strategies are more effective for plan generation.

5.3.2 ReAct: Interleaved Reasoning and Acting

The ReAct paradigm [Yao et al., 2022] represents a fundamental advance in LLM planning by interleaving reasoning traces with action execution. This approach enables agents to maintain reasoning coherence while interacting with external environments.

ReAct demonstrates significant improvements across diverse tasks: on question answering (HotpotQA) and fact verification (Fever), it overcomes hallucination issues prevalent in chain-of-thought reasoning by grounding reasoning in external knowledge. On interactive decision-making benchmarks (ALFWorld and WebShop), ReAct outperforms imitation and reinforcement learning baselines by 34% and 10% absolute success rates respectively.

The synergy between reasoning and acting enables ReAct to:

- Induce, track, and update action plans through reasoning traces
- Handle exceptions through reasoning-based error detection
- Interface with external sources for information gathering
- Generate interpretable, human-like task-solving trajectories

5.3.3 Tree-of-Thought Planning

Tree-of-Thought (ToT) planning extends chain-of-thought reasoning by exploring multiple reasoning paths through search algorithms. This approach enables systematic exploration of the planning space and supports backtracking when initial plans prove suboptimal.

The integration of ToT with multi-agent systems enables distributed exploration of planning branches, with different agents evaluating alternative decomposition strategies. This parallel exploration can significantly reduce planning time while maintaining solution quality.

5.3.4 Hierarchical Planning

Hierarchical planning operates across multiple levels of abstraction, enabling efficient reasoning about complex, long-horizon tasks. Ajay et al. [2023] introduce Compositional Foundation Models for Hierarchical Planning (HiP), which leverages multiple expert foundation models—language, vision, and action—to solve long-horizon robotic manipulation tasks.

HiP employs a three-level hierarchy:

1. **Strategic Level**: LLM-based symbolic plan construction

2. **Tactical Level**: Video diffusion model for visual plan grounding

3. **Operational Level**: Inverse dynamics model for action execution

Iterative refinement between levels ensures consistency, enabling effective transfer of abstract plans to concrete actions. This hierarchical decomposition reduces planning complexity while maintaining task coherence.

5.3.5 Search-Based Planning

Search-based approaches integrate classical planning algorithms with LLM reasoning. Chu et al. [2025] demonstrate that PDDL-based symbolic planners can complement LLM generation, providing logical correctness guarantees that pure neural approaches lack.

Their LLM+MAP framework for bimanual robot planning achieves superior performance compared to direct LLM planning (including GPT-4o, V3, and reasoning models o1 and R1) on metrics including planning time, success rate, and planning-step reduction. The structured representation enables efficient task allocation between robot arms while ensuring collision-free trajectories.

5.4 Execution Monitoring

Execution monitoring enables agents to track plan progress, detect deviations, and trigger corrective actions. Effective monitoring is essential for reliable operation in dynamic, uncertain environments.

5.4.1 Progress Estimation

Progress estimation provides agents with awareness of their advancement toward goals. Ma et al. [2019] introduce a self-monitoring agent for vision-and-language navigation that combines visual-textual co-grounding with progress monitoring. Their approach achieves an 8% absolute improvement in success rate on unseen environments by enabling agents to recognize when actions fail to advance toward goals.

The self-monitoring architecture comprises:

- **Visual-Textual Co-grounding**: Locates completed and pending instructions in observations
- **Progress Monitor**: Validates that grounded instructions reflect actual navigation progress
- **Decision Module**: Integrates grounding and progress signals for action selection

5.4.2 Hierarchical Reflection

Li et al. [2025b] propose hierarchical reflection for mobile task execution, enabling agents to self-monitor and recover from errors across multiple temporal scales. Their MobileUse system implements reflection-on-demand, triggering monitoring only when potential issues are detected to maintain efficiency.

The hierarchical reflection architecture operates at three levels:

1. **Action-Level**: Immediate detection of execution failures

2. **Task-Level**: Assessment of subtask completion

3. **Goal-Level**: Evaluation of overall objective achievement

This multi-scale monitoring enables graceful degradation and targeted recovery, with MobileUse achieving state-of-the-art success rates of 62.9% on AndroidWorld and 44.2% on AndroidLab benchmarks.

5.4.3 Interactive Monitoring for Agent Teams

Berry et al. [2003] present a framework for interactive execution monitoring of agent teams, addressing challenges in dynamic, data-rich domains with human oversight. Their Execution Assistant (EA) architecture customizes monitoring behavior for specific tasks, plans, and situations while respecting user preferences.

The framework categorizes monitoring alerts by urgency and type, using value-of-information principles to avoid operator overload. In evaluations across domains with hundreds of distributed agents (humans, robots, and vehicles), the EA achieved timely alert generation with less than 10% unwanted alerts as judged by domain experts.

5.5 Error Recovery Mechanisms

Error recovery enables agents to respond to execution failures, environmental changes, and unexpected outcomes. Robust recovery mechanisms are critical for deploying LLM-MAS in real-world applications.

5.5.1 Self-Correction Capabilities

Li [2025] systematically decompose LLM self-correction into three sub-capabilities: error detection, error localization, and error correction. Their cross-model experiments reveal an "Accuracy-Correction Paradox": weaker models achieve higher intrinsic correction rates than stronger models, suggesting that error depth—not just frequency—determines correctability.

Key findings include:

- Error detection rates vary dramatically (10% to 82%) across architectures
- Detection capability does not predict correction success
- Providing error location hints can hurt correction performance
- Stronger models make fewer but deeper errors resistant to self-correction

These insights challenge linear assumptions about model capability and self-improvement, with implications for designing self-refinement pipelines in multi-agent systems.

5.5.2 Hierarchical Task Decomposition for Recovery

Willibald and Lee [2025] propose learning hierarchical task decomposition from demonstrations specifically for execution monitoring and error recovery in robotic manipulation. Their unsupervised task segmentation algorithm combines intention recognition with feature clustering to infer task skills, enabling anomaly detection based on learned skill characteristics.

The framework supports incremental learning of recovery strategies as new situations arise, significantly reducing training data requirements compared to state-of-the-art learning techniques while efficiently learning complex in-contact behaviors.

5.5.3 Multi-Agent Error Recovery

In multi-agent settings, error recovery requires coordination among agents to redistribute tasks and adapt plans. Schwartz and Tokekar [2020] study robust multi-agent task assignment in failure-prone environments, presenting algorithms for both stochastic and adversarial failure models that yield optimal or near-optimal assignments while accounting for potential agent failures.

Key recovery strategies include:

- **Task Reallocation**: Reassigning failed tasks to available agents
- **Plan Adaptation**: Modifying coordination strategies
- **Role Reconfiguration**: Adjusting agent responsibilities
- **Escalation**: Requesting human intervention for unrecoverable errors

5.6 Multi-Agent Planning Coordination

Multi-agent planning coordination addresses the challenges of generating and executing plans that involve multiple agents with potentially conflicting objectives, partial observability, and communication constraints.

5.6.1 Centralized Planning, Decentralized Execution

The centralized planning with decentralized execution (CPDE) paradigm employs a central planner to generate coordinated plans that agents execute autonomously. This approach simplifies coordination during planning while enabling responsive execution.

Zhao et al. [2024] implement CPDE in their Hierarchical Auto-Organizing System (HAS) for multi-agent navigation in Minecraft. The system features:

- Hierarchical organization with centralized planning and decentralized execution
- Auto-organizing mechanisms for dynamic group adjustment
- Multi-modal perception integration for navigation tasks

The hierarchical structure enables efficient coordination while maintaining execution flexibility, demonstrating effective navigation in open-ended environments.

5.6.2 Distributed Planning

Distributed planning enables agents to collaboratively construct plans without central coordination. This approach enhances scalability and fault tolerance but introduces challenges in achieving global coherence.

Jo and Park [2025] investigate Byzantine-robust decentralized coordination of LLM agents, presenting consensus protocols that achieve agreement without leader dependency. Their approach enables robust coordination even when some agents behave maliciously or fail unexpectedly.

Pugachev [2025] introduce CodeCRDT for multi-agent code generation, employing Conflict-free Replicated Data Types (CRDTs) to enable lock-free concurrent editing. This approach achieves coordination through observation-driven merge strategies, enabling multiple agents to contribute to code generation without explicit synchronization.

5.6.3 Negotiation-Based Planning

Negotiation-based planning enables agents to resolve conflicts and reach agreements through structured communication. **?** demonstrate that augmenting agent action spaces with conventions—predefined cooperative action sequences—significantly improves multi-agent coordination in partially observable environments like Hanabi.

Conventions enable implicit knowledge sharing without explicit communication, reducing coordination overhead while improving performance. The approach achieves substantial improvements in both self-play and cross-play scenarios across varying numbers of cooperating agents.

5.6.4 Plan Reuse and Transfer

Plan reuse enables agents to leverage previously successful plans for similar tasks, reducing planning latency. Li et al. [2025a] present AgentReuse, a mechanism that achieves 93% effective plan reuse rate through intent classification for similarity evaluation.

Key components of AgentReuse include:

- Intent classification for semantic similarity assessment
- Plan retrieval based on request characteristics
- Plan adaptation for context-specific requirements

The mechanism reduces planning latency by 93.12% compared to baselines without reuse, demonstrating the value of experiential learning in planning systems.

5.7 Single-Agent vs. Collaborative Planning

The distinction between single-agent and collaborative planning extends beyond the number of agents involved to encompass fundamental differences in planning paradigms, coordination requirements, and solution characteristics.

5.7.1 Single-Agent Planning Characteristics

Single-agent planning focuses on individual task completion with full control over action selection. Key characteristics include:

- **Simplified State Space**: No need to model other agents' intentions
- **Direct Optimization**: Plans optimize single objective function
- **No Coordination Overhead**: All decisions made locally
- **Full Observability Assumption**: Agent has complete task knowledge

However, single-agent planning may struggle with tasks requiring diverse expertise or parallel execution. Shen et al. [2024] demonstrate that decomposing capabilities across multiple specialized agents (planner, caller, summarizer) outperforms monolithic single-agent approaches, particularly for smaller LLMs.

5.7.2 Collaborative Planning Advantages

Collaborative planning enables:

- **Distributed Expertise**: Specialized agents handle domain-specific subtasks
- **Parallel Execution**: Concurrent action execution reduces completion time
- **Redundancy**: Multiple agents provide fault tolerance
- **Diverse Perspectives**: Different reasoning approaches improve solution quality

Xiao et al. [2024] demonstrate collaborative planning in financial trading, where specialized agents (fundamental analysts, sentiment analysts, technical analysts) contribute diverse perspectives to trading decisions. The framework includes bull and bear researchers for balanced market assessment, achieving superior returns, Sharpe ratio, and maximum drawdown compared to baseline models.

5.7.3 Trade-offs and Design Considerations

The choice between single-agent and collaborative planning involves trade-offs:

Orogat et al. [2026] provide empirical analysis through MAFBench, demonstrating that framework-level design choices can increase latency by over 100x, reduce planning accuracy by up to 30%, and lower coordination success from above 90% to below 30%. These findings highlight the critical importance of architectural decisions in multi-agent planning systems.

5.8 Integration with Memory Systems

Memory systems play a crucial role in planning by enabling agents to leverage past experiences, maintain task context, and accumulate procedural knowledge.

5.8.1 Procedural Memory for Planning

Forouzandeh et al. [2025] present MACLA, a framework that decouples reasoning from learning by maintaining a frozen LLM with external hierarchical procedural memory. MACLA extracts reusable procedures from trajectories, tracks reliability through Bayesian posteriors, and refines procedures through contrastive learning.

Across four benchmarks (ALFWorld, WebShop, TravelPlanner, InterCodeSQL), MACLA achieves 78.1% average performance, constructing memory in 56 seconds—2800 times faster than parameter-training baselines. The system compresses 2851 trajectories into 187 procedures, demonstrating efficient knowledge consolidation.

5.8.2 Memory Types in Planning

Different memory types serve distinct planning functions:

- **Working Memory**: Maintains current task context and active goals
- **Episodic Memory**: Stores past experiences for case-based reasoning
- **Semantic Memory**: Contains domain knowledge and facts
- **Procedural Memory**: Encodes skills and action sequences

Ye [2025] introduce the Task Memory Engine (TME) for enhancing state awareness in multi-step LLM agent tasks. TME maintains structured task representations that enable agents to track progress and resume interrupted tasks effectively.

5.9 Open Challenges and Future Directions

Despite significant advances, planning in LLM-MAS faces several open challenges:

5.9.1 Long-Horizon Planning

Planning over extended time horizons remains challenging due to compounding errors and the difficulty of maintaining coherent long-term goals. Aghzal et al. [2026] analyze LLM web agents from a hierarchical planning perspective, finding that while structured PDDL plans produce more concise strategies, low-level execution remains the dominant bottleneck.

Addressing long-horizon planning requires:

- Hierarchical abstractions that manage complexity
- Robust execution monitoring with timely intervention
- Adaptive replanning capabilities
- Memory systems that maintain goal coherence

5.9.2 Scalability

Planning coordination overhead grows with the number of agents, challenging scalability to large multi-agent systems. Sonu et al. [2015] exploit anonymity and context-specific independence to achieve computational gains, enabling planning for systems with over 1,000 agents.

Future research must address:

- Efficient coordination protocols for large agent populations
- Approximate planning methods with bounded sub-optimality
- Hierarchical decomposition for scalable coordination

5.9.3 Safety and Alignment

Planning agents may generate plans that achieve goals through unintended or harmful means. Chen et al. [2026] identify that agents are "too helpful to be safe," bypassing safety constraints in over 92% of cases when users do not explicitly request safety checks.

Ensuring safe planning requires:

- Constitutional constraints embedded in planning processes
- Explicit task boundaries and stopping rules
- Human oversight integration
- Value alignment across planning objectives

5.9.4 Formal Verification

Integrating formal verification with LLM-based planning remains an open challenge. While symbolic planners provide correctness guarantees, neural approaches lack formal verification methods. Hybrid approaches that combine LLM reasoning with formal verification represent a promising direction.

5.10 Summary

Planning in LLM-based multi-agent systems encompasses a rich landscape of techniques spanning task decomposition, plan generation, execution monitoring, and error recovery. Key insights from this analysis include:

1. **Decomposition Diversity**: Multiple decomposition paradigms (goal-based, HTN, role-based, temporal) address different planning requirements, with hybrid approaches combining their strengths.
2. **Reasoning-Action Integration**: The ReAct paradigm demonstrates the power of interleaving reasoning with action, enabling more robust and interpretable planning.
3. **Hierarchical Organization**: Multi-level planning hierarchies enable efficient reasoning about complex tasks while maintaining execution flexibility.
4. **Monitoring Criticality**: Execution monitoring is essential for reliable operation, with hierarchical reflection enabling multi-scale error detection and recovery.
5. **Coordination Trade-offs**: The choice between centralized and distributed planning involves fundamental trade-offs between coordination efficiency and system robustness.
6. **Memory Integration**: External memory systems significantly enhance planning capabilities by enabling experience accumulation and procedural knowledge reuse.

Future research directions include developing more robust long-horizon planning methods, scalable coordination protocols for large agent populations, and formal verification techniques for neural planning approaches. The integration of symbolic planning guarantees with neural reasoning flexibility represents a particularly promising direction for advancing planning capabilities in LLM-MAS.

6 Evaluation Methodologies

The evaluation of LLM-based multi-agent systems (LLM-MAS) presents unique challenges that extend beyond traditional single-agent or non-agentic LLM evaluation. In this section, we provide a comprehensive analysis of evaluation methodologies, covering benchmark datasets, evaluation metrics, evaluation frameworks, and reproducibility challenges specific to multi-agent settings.

6.1 Benchmark Datasets

Benchmark datasets serve as the foundation for systematic evaluation and comparison of LLM-MAS. We categorize existing benchmarks along two primary dimensions: *evaluation objectives* (task performance, coordination quality, safety) and *domain specificity* (general-purpose vs. domain-specific).

6.1.1 General-Purpose Multi-Agent Benchmarks

MAFBench. Orogat et al. [2026] introduce MAFBench, a unified evaluation suite designed specifically for multi-agent LLM frameworks. MAFBench addresses the critical gap in framework-level evaluation by integrating existing benchmarks under a standardized execution pipeline. The benchmark evaluates multiple capabilities including orchestration overhead, memory behavior, planning accuracy, and coordination success. Empirical studies using MAFBench reveal that framework-level design choices alone can increase latency by over 100×, reduce planning accuracy by up to 30%, and lower coordination success from above 90% to below 30%.

DevAI Benchmark. Introduced alongside the Agent-as-a-Judge framework [Zhuge et al., 2024], DevAI provides 55 realistic AI development tasks requiring multi-step planning, tool use, and information retrieval. Tasks range from simple script generation to complex system design, enabling evaluation of both individual agent capabilities and inter-agent coordination. The benchmark includes intermediate milestones that allow for partial credit evaluation and process-aware assessment.

ZSC-Eval. **?** present ZSC-Eval, the first evaluation toolkit specifically designed for zero-shot coordination in multi-agent systems. The benchmark addresses the unique challenge of evaluating agents that must collaborate with unseen partners. ZSC-Eval introduces the Best-Response Proximity (BR-Prox) metric and uses behavior-preferring rewards to generate diverse evaluation partner candidates, enabling more robust assessment of coordination generalization.

6.1.2 Domain-Specific Benchmarks

Code Generation and Software Engineering. HumanEvalComm [Wu and Fard, 2025] extends traditional code generation benchmarks to evaluate communication competence in multi-agent code development scenarios. The benchmark assesses not only code correctness but also the quality of inter-agent communication during collaborative programming tasks. PPTC and PPTC-R [Guo et al., 2023, Zhang et al., 2024b] focus on multi-turn, multi-modal task completion for office automation, evaluating agents' ability to handle PowerPoint task instructions with robustness to adversarial inputs and version variations.

Game Playing and Strategic Reasoning. Avalon-Bench [Light et al., 2023] evaluates LLM agents in the social deduction game Avalon, requiring strategic reasoning, deception detection, and team coordination. The benchmark reveals significant challenges in Theory of Mind reasoning and long-term strategic planning. ALYMPICS [Mao et al., 2023] introduces game-theoretic scenarios including water allocation challenges and auction dynamics, testing agents' ability to reason about other agents' strategies and reach optimal outcomes.

Scientific and Technical Domains. GeoBenchX [Krechetova and Kochedykov, 2025] provides multi-step geospatial reasoning tasks that require agents to decompose complex spatial queries and coordinate information retrieval across multiple data sources. CORE-Bench [**?**] focuses on computational reproducibility, with 270 tasks based on 90 scientific papers across computer science, social science, and medicine, evaluating agents' ability to reproduce computational results.

Security and Safety. ZeroDayBench [Lau et al., 2026] evaluates LLM agents on detecting zero-day vulnerabilities, a critical capability for autonomous security systems. The benchmark tests both individual agent detection capabilities and multi-agent verification protocols, revealing that current systems achieve only 21% accuracy on the hardest tasks.

6.2 Evaluation Metrics

Effective evaluation of LLM-MAS requires metrics that capture multiple dimensions of system behavior. We organize metrics into three primary categories: task performance metrics, coordination quality metrics, and system property metrics.

6.2.1 Task Performance Metrics

Task performance metrics quantify the effectiveness of multi-agent systems in achieving their objectives.

Success Metrics.

- **Success Rate**: The proportion of tasks completed successfully, defined as $\frac{\#\text{success}}{\#\text{total}}$. This fundamental metric provides a coarse-grained view of system capability but may obscure nuanced performance differences.
- **Completion Rate**: The proportion of tasks fully completed, distinguishing between partial and complete task achievement.
- **Partial Success Score**: A continuous score $\in [0, 1]$ capturing the degree of goal achievement, particularly relevant for complex, multi-objective tasks.

Table 5: Comparison of Evaluation Benchmarks for LLM-Based Multi-Agent Systems

Benchmark	Domain	Tasks	Multi-Agent	Process Eval.	Year
MAFBench	General	Multiple	✓	✓	2026
DevAI	Software Dev.	55	✓	✓	2024
ZSC-Eval	Coordination	6+ envs	✓	✓	2023
HumanEvalComm	Code Gen.	164+	✓	–	2024
AvalonBench	Game Playing	15+	✓	–	2023
CORE-Bench	Reproducibility	270	✓	✓	2024
ZeroDayBench	Security	100+	✓	–	2026
PPTC/PPTC-R	Office Tasks	28	Partial	–	2023/24
GeoBenchX	Geospatial	50+	✓	–	2025

Quality Metrics.

- **Accuracy**: Correctness of outputs against ground truth, applicable to classification and prediction tasks.
- **F1 Score**: Harmonic mean of precision and recall, particularly relevant for information retrieval and extraction tasks.
- **BLEU/ROUGE**: N-gram overlap metrics for text generation quality, though limited in capturing semantic equivalence.

Efficiency Metrics.

- **Time to Completion**: Wall-clock time from task initiation to completion, capturing real-world deployability.
- **Steps to Completion**: Number of actions or turns required, relevant for planning efficiency.
- **Token Efficiency**: LLM token consumption per task, directly impacting operational costs.

6.2.2 Coordination Quality Metrics

Coordination quality metrics capture the effectiveness of inter-agent interaction and collaboration.

Communication Metrics.

- **Message Count**: Total messages exchanged between agents, with lower values typically indicating more efficient coordination.
- **Communication Rate**: Messages per task or per unit time, normalized for task complexity.
- **Good Question Rate**: The proportion of information-seeking messages that yield useful responses, introduced in LLM-Coordination [**?**].
- **Redundancy Ratio**: Proportion of messages containing duplicate or unnecessary information, indicating potential coordination inefficiencies.

Consensus Metrics.

- **Agreement Rate**: Proportion of decisions where agents reach consensus, relevant for voting and negotiation protocols.
- **Convergence Time**: Number of iterations or time required to reach consensus.
- **Consensus Quality**: Correctness of agreed-upon decisions, distinguishing between rapid but incorrect consensus and slower but accurate agreement.

Cooperation Metrics.

- **Win Rate**: Team success rate in competitive or cooperative game scenarios.
- **Cooperation Score**: Composite metric capturing effective collaboration behaviors, often domain-specific.
- **Role Adherence**: Degree to which agents follow assigned roles and responsibilities.

6.2.3 System Property Metrics

System property metrics assess non-functional characteristics critical for real-world deployment.

Scalability Metrics.

- **Agent Scaling Factor**: Performance degradation as the number of agents increases, typically measured as latency increase or accuracy decrease per additional agent.
- **Communication Overhead**: Growth rate of inter-agent communication as system size increases, often following $O(n^2)$ patterns in fully-connected architectures.

Robustness Metrics.

- **Fault Tolerance**: Performance retention when agents fail or behave unexpectedly.
- **Adversarial Robustness**: Performance under adversarial conditions, including malicious agents or poisoned inputs.
- **Graceful Degradation**: Rate of performance decay as system stress increases.

Safety and Alignment Metrics.

- **Constraint Violation Rate**: Frequency of safety constraint violations during task execution.
- **Constitutional Compliance**: Adherence to predefined behavioral rules [de Curtò and de Zarzà, 2026].
- **Hallucination Rate**: Frequency of factually incorrect or fabricated information in agent outputs.

6.3 Evaluation Frameworks

Evaluation frameworks provide structured methodologies for conducting comprehensive assessments of LLM-MAS. We review several prominent frameworks and their distinctive approaches.

6.3.1 Agent-as-a-Judge

Zhuge et al. [2024] propose the Agent-as-a-Judge paradigm, which leverages LLM agents themselves to evaluate other agents. This approach addresses the challenge of evaluating open-ended, complex tasks where traditional metrics are insufficient. The framework introduces:

- **Intermediate Feedback**: Agents provide feedback at multiple stages of task execution, enabling process-aware evaluation rather than solely outcome-based assessment.
- **Multi-Agent Evaluation Panels**: Multiple evaluator agents assess different aspects of performance, reducing individual bias through ensemble judgment.
- **Structured Evaluation Criteria**: Predefined rubrics guide agent evaluators, improving consistency and interpretability.

The Agent-as-a-Judge framework demonstrates strong correlation with human evaluations on the DevAI benchmark, while significantly reducing evaluation costs compared to human annotation.

6.3.2 AEMA: Verifiable Evaluation Framework

Lee et al. [2026] introduce AEMA (Auditable and Explainable Multi-agent Assessment), a framework designed for trustworthy and controlled evaluation of agentic LLM systems. Key features include:

- **Process-Aware Evaluation**: Tracking and assessing intermediate reasoning steps, not just final outputs.
- **Audit Trails**: Comprehensive logging of agent actions, decisions, and communications for post-hoc analysis.
- **Controlled Environments**: Sandboxed execution environments that enable reproducible evaluation while preventing unintended side effects.

AEMA addresses the challenge of evaluating agents in high-stakes domains where process integrity is as important as outcome quality.

6.3.3 The Social Laboratory

Reza [2025] propose a psychometric framework for multi-agent evaluation, drawing on established psychological measurement theory. The framework introduces:

- **Psychometric Validity**: Applying concepts of construct validity, reliability, and fairness from psychometrics to agent evaluation.
- **Consensus-Seeking Protocols**: Structured debate and discussion protocols that reveal agent reasoning processes.

- **Behavioral Assessment**: Evaluating agents through observable behaviors rather than self-reported capabilities.

This framework is particularly relevant for evaluating multi-agent debate systems and collaborative decision-making scenarios.

6.3.4 TED Framework

? introduce the TED (Talk, Evaluate, Diagnose) framework, which emphasizes user-aware evaluation:

- **User Persona Templates**: Reusable templates for expert and non-expert user interactions, enabling systematic evaluation across user expertise levels.
- **Subgoal-Based Assessment**: Breaking complex tasks into subgoals with natural language grading notes.
- **Automated Error Analysis**: Systematic identification and categorization of agent errors, providing actionable feedback for improvement.

The TED framework reveals that agent performance varies significantly across user expertise levels, highlighting the importance of user-aware evaluation design.

6.3.5 Evaluation Agent

Zhang et al. [2024a] propose the Evaluation Agent framework for efficient evaluation of visual generative models. While originally designed for single-system evaluation, its principles extend to multi-agent settings:

- **Human-Like Evaluation Strategies**: Mimicking human expert evaluation patterns for efficient, multi-round assessment.
- **Promptable Evaluation**: User-tailored evaluation criteria that adapt to specific needs.
- **Efficiency Gains**: Reducing evaluation time to 10% of traditional methods while maintaining comparable accuracy.

6.4 Reproducibility Challenges

Reproducibility poses significant challenges for LLM-MAS evaluation, threatening the validity of comparative studies and the reliability of reported results.

6.4.1 Sources of Non-Determinism

LLM Stochasticity. Large language models exhibit inherent stochasticity in their outputs, even with fixed random seeds. Temperature settings, sampling strategies, and model version differences all contribute to output variability. In multi-agent settings, this stochasticity compounds across multiple agents, leading to exponential growth in the space of possible interaction trajectories.

Agent Interaction Dynamics. Multi-agent systems exhibit emergent behaviors that arise from complex interaction patterns. Small variations in initial conditions or individual agent behaviors can cascade into substantially different system-level outcomes, a phenomenon analogous to the butterfly effect in complex systems.

Environmental Factors. External dependencies, including API availability, network latency, and service version changes, introduce additional variability. The temporal dynamics of real-world systems further complicate reproducibility, as agent behaviors may change based on external state.

6.4.2 Framework-Level Reproducibility

Orogat et al. [2026] identify significant reproducibility challenges at the framework level:

- **Architectural Variations**: Different frameworks impose different architectural constraints that affect agent behavior, making cross-framework comparisons challenging.
- **Implementation Differences**: Variations in message passing, memory management, and execution scheduling can lead to divergent behaviors even with identical agent configurations.
- **Evaluation Pipeline Inconsistencies**: Lack of standardized evaluation protocols leads to incomparable results across studies.

6.4.3 Mitigation Strategies

Standardized Protocols. The development of standardized evaluation protocols, such as those provided by MAFBench, enables more reproducible comparisons. These protocols specify:

- Fixed random seeds and deterministic generation settings

- Standardized task formulations and success criteria
- Consistent logging and reporting formats

Multiple Run Aggregation. Conducting multiple evaluation runs with statistical aggregation (mean, standard deviation, confidence intervals) provides more robust estimates of system performance. However, the computational cost of multiple runs can be prohibitive for large-scale multi-agent systems.

Controlled Environments. Sandboxed execution environments, as implemented in AEMA, isolate evaluation from external variability sources. These environments control for API availability, network conditions, and service versions.

Comprehensive Reporting. Detailed reporting of experimental conditions, including model versions, hyperparameters, and environmental configurations, enables more meaningful interpretation and potential reproduction of results. The adoption of standardized reporting templates, similar to model cards for single LLMs, would benefit the multi-agent community.

6.4.4 Benchmark-Specific Reproducibility Considerations

CORE-Bench. **?** specifically address reproducibility in scientific research contexts. The benchmark provides a parallelizable evaluation system that reduces evaluation time while maintaining consistency, demonstrating that efficient evaluation need not compromise reproducibility.

ZSC-Eval. The zero-shot coordination evaluation framework introduces Best-Response Diversity (BR-Div) for selecting evaluation partners, ensuring that coordination assessments are robust to partner selection variability while remaining reproducible across evaluation runs.

6.5 Comparative Analysis of Evaluation Approaches

We present a comparative analysis of evaluation approaches across key dimensions relevant to LLM-MAS research and deployment.

6.5.1 Trade-offs in Evaluation Design

Comprehensiveness vs. Efficiency. Comprehensive evaluation frameworks that assess multiple dimensions (task performance, coordination quality, safety) provide richer insights but require greater computational resources and time. The Evaluation Agent framework demonstrates that efficiency can be improved through intelligent sampling strategies without significant accuracy loss.

Automation vs. Human Alignment. Fully automated evaluation using LLM-as-judge approaches enables scalable assessment but may diverge from human preferences. Hybrid approaches that combine automated evaluation with selective human verification offer a middle ground.

Standardization vs. Flexibility. Standardized benchmarks enable cross-study comparisons but may not capture domain-specific requirements. Domain-specific benchmarks provide relevant assessment but limit generalizability. The balance between standardization and flexibility remains an active area of research.

6.5.2 Emerging Evaluation Paradigms

Continuous Evaluation. As multi-agent systems are deployed in dynamic environments, continuous evaluation during operation becomes necessary. This paradigm requires lightweight evaluation mechanisms that can assess system performance without disrupting normal operation.

Human-in-the-Loop Evaluation. Integrating human feedback into the evaluation loop provides direct alignment with human preferences but introduces scalability challenges. Effective human-in-the-loop evaluation requires careful consideration of when and how to solicit human input.

Adversarial Evaluation. Evaluating systems under adversarial conditions reveals robustness limitations that may not be apparent under normal operation. Frameworks like ZeroDayBench exemplify this approach, testing systems against worst-case scenarios.

Table 7: Comparative Analysis of Evaluation Frameworks

Framework	Process Eval.	Multi-Agent	User-Aware	Reproducibility	Efficiency
Agent-as-a-Judge	✓	✓	–	Medium	High
AEMA	✓	✓	–	High	Medium
Social Laboratory	✓	✓	–	Medium	Low
TED	✓	✓	✓	Medium	Medium
Evaluation Agent	–	–	–	High	High
MAFBench	✓	✓	–	High	Medium

6.6 Future Directions

The evaluation of LLM-based multi-agent systems remains an evolving field with several promising research directions:

1. **Unified Benchmark Development**: The community would benefit from consensus on standardized benchmark suites that enable meaningful cross-study comparisons while remaining adaptable to domain-specific requirements.

2. **Process-Aware Metrics**: Development of metrics that capture the quality of reasoning processes, not just outcomes, is essential for evaluating systems intended for high-stakes applications.

3. **Scalable Human Alignment**: Techniques for efficiently aligning automated evaluation with human preferences, potentially through active learning or preference modeling, would improve the validity of large-scale evaluations.

4. **Emergent Behavior Detection**: Methods for systematically identifying and characterizing emergent behaviors in multi-agent systems would improve our understanding of these complex systems.

5. **Reproducibility Infrastructure**: Development of shared infrastructure for reproducible evaluation, including standardized environments, logging protocols, and reporting templates, would accelerate research progress.

In summary, the evaluation of LLM-based multi-agent systems requires a multi-faceted approach that considers task performance, coordination quality, and system properties. While significant progress has been made in developing benchmarks and frameworks, challenges in reproducibility, standardization, and scalability remain active areas of research. The continued development of rigorous evaluation methodologies is essential for advancing the field and enabling reliable deployment of multi-agent systems in real-world applications.

7 Cross-Domain Applications

The versatility of LLM-based multi-agent systems has enabled their deployment across a diverse range of application domains. This section examines how the architectural patterns, coordination mechanisms, and planning strategies discussed in previous sections manifest in real-world applications, highlighting both the successes and domain-specific challenges that have emerged.

7.1 Code Generation and Software Development

Software development represents one of the most successful application domains for LLM-based multi-agent systems, owing to the natural alignment between software engineering workflows and agent collaboration patterns.

7.1.1 End-to-End Software Development

ChatDev [Qian et al., 2024] pioneered the paradigm of using communicative agents for comprehensive software development. The framework organizes agents into specialized roles mirroring real software teams—including Chief Executive Officer, Chief Product Officer, Chief Technology Officer, programmers, and testers—who collaborate through structured dialogues. ChatDev introduces two key innovations: (1) a *chat chain* that guides agents through sequential phases (design, coding, testing), and (2) *communicative dehallucination* mechanisms that reduce code hallucinations through iterative refinement. The framework demonstrates that natural language communication effectively facilitates system design, while programming language communication proves advantageous for debugging tasks.

MetaGPT [?] extends this paradigm by incorporating Standardized Operating Procedures (SOPs) from software engineering practice. The framework enforces structured workflows including requirement analysis, system design, and implementation, with agents producing artifacts such as Product Requirement Documents (PRDs) and system architecture diagrams before code generation. This approach addresses the common failure mode where LLM agents generate plausible but architecturally unsound code.

7.1.2 Multi-Agent Code Collaboration

Recent work has explored more sophisticated collaboration patterns for code generation. **CodeCRDT** [Pugachev, 2025] introduces Conflict-free Replicated Data Types (CRDTs) for lock-free concurrent code generation, enabling multiple agents to work on the same codebase simultaneously without coordination overhead. This approach achieves significant improvements in code generation throughput while maintaining consistency.

Context Engineering for Multi-Agent LLM Code Assistants [Haseeb, 2025] demonstrates a workflow combining intent clarification, retrieval-augmented generation, and specialized sub-agents orchestrated via Claude's agent framework. The system achieves higher single-shot success rates and better adherence to project context compared to baseline single-agent approaches, particularly for complex multi-file repositories.

7.1.3 Software Testing and Quality Assurance

Multi-agent systems have also shown promise in software testing. **AutoHarness** [Lou et al., 2026] synthesizes test harnesses automatically, while **HumanEval-Comm** [?] introduces a benchmark specifically evaluating communication competence in code generation, highlighting the importance of inter-agent communication quality for collaborative programming tasks.

7.2 Scientific Discovery and Research Automation

The application of LLM-based multi-agent systems to scientific discovery represents a paradigm shift from AI as a tool to AI as a research collaborator, capable of autonomously conducting experiments, analyzing results, and generating novel hypotheses.

7.2.1 Autonomous Scientific Research

Jr. AI Scientist [?] represents a state-of-the-art autonomous AI scientist system that mimics the core research workflow of a novice student researcher. Given a baseline paper, the system analyzes limitations, formulates novel hypotheses, iteratively experiments until improvements are achieved, and writes a paper with results. The system successfully generated new research papers building upon real NeurIPS, IJCV, and ICLR works by proposing and implementing novel methods.

Aster [?] achieves autonomous scientific discovery over 20 times faster than existing frameworks through iterative program improvement. Applied to problems in mathematics, GPU kernel engineering, biology, neuroscience, and language model training, Aster attains state-of-the-art results in every task except one, where it matches the performance of the best human solution with less than 1/190th of the compute.

7.2.2 Domain-Specific Scientific Applications

MASTER [Rothfarb et al., 2025] applies hierarchical multi-agent LLM reasoning to autonomous functional materials discovery. The framework employs active learning for atomistic simulations, with agents coordinating to propose candidate materials, execute density functional theory (DFT) calculations, and iteratively refine search strategies based on results.

Lang-PINN [He et al., 2025] introduces a multi-agent framework for physics-informed neural networks, enabling automated solving of partial differential equations. Multiple agents handle different aspects: problem formulation, network architecture design, training supervision, and result verification.

7.2.3 Research Infrastructure and Benchmarks

The emergence of AI-driven scientific discovery has prompted development of supporting infrastructure. **Agentic Science** [?] provides a comprehensive survey positioning autonomous scientific discovery within the broader AI for Science paradigm, identifying five core capabilities underpinning scientific agency: hypothesis generation, experimental design, execution, analysis, and iterative refinement.

7.3 Game Playing and Strategic Reasoning

Game environments provide controlled yet challenging testbeds for evaluating multi-agent coordination, strategic reasoning, and emergent behaviors in LLM-based systems.

7.3.1 Social Deduction Games

AvalonBench [Light et al., 2023] evaluates LLMs playing the game of Avalon, a social deduction game requiring strategic reasoning, deception detection, and coalition building. The benchmark reveals both the capabilities and limitations of current LLM agents in handling complex social dynamics with hidden information and adversarial players.

ALYMPICS [Mao et al., 2023] explores strategic decision-making through game-theoretic scenarios including water allocation challenges and auction dynamics. The framework demonstrates how LLM agents can engage in economic reasoning and strategic behavior, though with notable gaps in optimal play compared to game-theoretic solutions.

7.3.2 Cooperative and Competitive Games

Multi-Agent Training for Pommerman [Huynh et al., 2024] combines curriculum learning with population-based self-play for training agents in the Pommerman environment—a complex multi-agent game requiring both cooperation and competition. The hybrid LLM-MARL approach demonstrates improved performance over pure MARL baselines.

Do LLM Agents Have Regret? [?] analyzes LLM agents in online learning and game settings, revealing that while LLMs can achieve no-regret learning in some scenarios, they often fail to maintain consistent strategies across repeated plays, highlighting fundamental challenges in strategic reasoning.

7.3.3 Game Balancing and Content Generation

RuleSmith [Zeng et al., 2026] applies multi-agent LLMs to automated game balancing, demonstrating how agents can collaboratively analyze game mechanics, identify imbalance issues, and propose rule modifications. This application showcases the creative potential of multi-agent systems beyond gameplay itself.

7.4 Robotics and Embodied Agents

Embodied AI applications present unique challenges for multi-agent systems, requiring integration of perception, action, and physical world reasoning.

7.4.1 Task Planning for Robotics

LLM+MAP [Chu et al., 2025] integrates Large Language Models with Planning Domain Definition Language (PDDL) for bimanual robot task planning. The framework leverages LLMs for high-level task decomposition while using classical planners for constraint satisfaction, demonstrating effective coordination between two robotic arms in complex manipulation tasks.

From Language to Action [Shaji et al., 2026] investigates whether LLM-based agents can serve as cognitive controllers for embodied robot cognition. The study reveals that while LLMs can effectively translate natural language instructions into action sequences, challenges remain in handling physical uncertainties and real-time execution requirements.

7.4.2 Multi-Robot Coordination

ADMM-Based Distributed MPC [?] proposes a fully decentralized model predictive control framework with control barrier function constraints for multi-robot quadrupedal locomotion. The approach enables fully decentralized trajectory optimization with symmetric computational load across agents, achieving performance comparable to centralized MPC while reducing per-cycle planning time by up to 51%.

Embodied-RAG [Xie et al., 2024] introduces non-parametric embodied memory for navigation, enabling agents to retrieve and generate based on past experiences without explicit map representations. This approach demonstrates improved generalization to novel environments compared to traditional SLAM-based methods.

7.4.3 Vision-Language-Action Models

Vision-Language-Action (VLA) models represent an emerging paradigm integrating perception, language understanding, and motor control. **FASTER** [?] rethinks real-time flow VLAs, introducing horizon-aware scheduling that adaptively prioritizes near-term actions during flow sampling. Real-world experiments including a dynamic table tennis task demonstrate unprecedented real-time responsiveness for generalist policies.

OmniVTA [?] presents a world-model-based visuo-tactile manipulation framework for contact-rich robotic tasks. The system integrates self-supervised tactile encoding, a two-stream visuo-tactile world model, and a 60Hz reflexive controller for closed-loop tactile feedback, demonstrating strong generalization to unseen objects and configurations.

7.5 Financial and Economic Applications

Financial domains present unique requirements for multi-agent systems, including real-time decision-making, risk management, and handling of heterogeneous information sources.

7.5.1 Trading and Investment

TradingAgents [Xiao et al., 2024] introduces a multi-agent LLM framework inspired by real-world trading firm structures. The system employs a manager-analyst hierarchy with specialized agents for different market analysis tasks, including bull and bear researchers who provide opposing viewpoints on investment decisions.

FinCon [Yu et al., 2024] proposes a synthesized LLM multi-agent system with conceptual verbal reinforcement for financial decision making. The framework incorporates a risk-control component that episodically initiates self-critiquing to update systematic investment beliefs, achieving strong generalization across single stock trading and portfolio management tasks.

7.5.2 Financial Analysis and Reasoning

FinTradeBench [?] introduces a benchmark for evaluating financial reasoning that integrates company fundamentals and trading signals. The benchmark contains 1,400 questions grounded in NASDAQ-100 companies over a ten-year historical window, organized into fundamentals-focused, trading-signal-focused, and hybrid reasoning categories.

7.6 Healthcare and Medical Applications

Healthcare applications of multi-agent LLM systems require careful attention to accuracy, safety, and the integration of domain-specific knowledge.

MedAide [Yang et al., 2024a] presents an information fusion framework for medical intents via LLM-based agent collaboration. The system employs rotation collaboration patterns where different agents specialize in various medical domains, with an orchestrating agent synthesizing responses. The framework demonstrates improved diagnostic accuracy through multi-perspective analysis while maintaining medical safety constraints.

Caging the Agents [?] proposes a zero-trust security architecture specifically designed for healthcare multi-agent systems, addressing the unique privacy and safety requirements of medical applications.

7.7 Security and Cyber Defense

Multi-agent LLM systems have shown promise in security applications, where the ability to coordinate multiple specialized agents can enhance detection and response capabilities.

7.7.1 Threat Detection

PhishDebate [Li et al., 2025c] introduces a multi-agent debate framework for phishing website detection. Multiple agents with different detection strategies engage in structured debate, with a judge agent synthesizing the final verdict. The debate mechanism improves detection accuracy while providing interpretable reasoning traces.

ZeroDayBench [Lau et al., 2026] provides a benchmark for evaluating LLM agents on zero-day vulnerability detection for cyber defense. The benchmark tests agents' ability to identify previously unseen vulnerabilities, a critical capability for proactive security.

7.7.2 Claim Verification

Thucy [Theologitis and Suciu, 2025] presents an LLM-based multi-agent system for claim verification across relational databases. The system coordinates multiple agents for claim decomposition, evidence retrieval, and verification, demonstrating improved accuracy over single-agent baselines in cross-database fact-checking scenarios.

7.8 Other Emerging Applications

7.8.1 Urban Planning

Planning, Living and Judging [?] proposes a cyclical framework for urban planning where agents alternate between planning, simulation, and evaluation roles. The multi-agent approach enables consideration of multiple stakeholder perspectives and iterative refinement of urban development proposals.

7.8.2 Innovation and Problem Solving

TRIZ Agents [?] applies multi-agent LLMs to TRIZ-based innovation, a structured methodology for inventive problem solving. Agents with different domain expertise collaborate to navigate TRIZ steps, demonstrating the potential for AI-driven innovation in engineering design.

7.8.3 Content Creation and Entertainment

CompanionCast [?] explores social collaboration with multi-agent systems for shared entertainment experiences, demonstrating how coordinated agents can enhance content consumption and social interaction.

7.9 Cross-Domain Patterns and Insights

Analysis across application domains reveals several consistent patterns:

Hierarchical Architectures Dominate: Most successful applications employ hierarchical organization patterns, with manager-coordinator agents orchestrating specialized worker agents. This pattern appears consistently across code generation (ChatDev, MetaGPT), financial trading (TradingAgents, FinCon), and scientific discovery (Jr. AI Scientist).

Domain Knowledge Integration: Effective systems incorporate domain-specific knowledge through retrieval-augmented generation, specialized prompting, or fine-tuning. The importance of domain knowledge is particularly evident in healthcare (MedAide) and scientific discovery (MASTER, Lang-PINN) applications.

Communication Protocol Variation: Different domains require different communication patterns. Code generation benefits from structured, phase-based communication; game playing requires real-time coordination; and scientific discovery demands iterative refinement loops.

Safety and Verification Requirements: Applications in healthcare, finance, and security share requirements for safety constraints and verification mechanisms, often implemented through multi-agent debate or consensus-seeking procedures.

The diversity of successful applications demonstrates the flexibility of the LLM-based multi-agent paradigm while highlighting domain-specific adaptations necessary for practical deployment. As the field matures, we anticipate further specialization of architectures and coordination mechanisms tailored to the unique requirements of each application domain.

8 Open Challenges and Future Research Directions

The rapid advancement of LLM-based multi-agent systems has unveiled remarkable capabilities in coordination, planning, and task execution. However, this progress has also exposed significant challenges that must be addressed before these systems can achieve widespread deployment in critical applications. This section synthesizes insights from our comprehensive review to identify the most pressing open problems and outline promising research directions. We organize our discussion around six fundamental dimensions: scalability, robustness, interpretability, safety and alignment, evaluation, and theoretical foundations.

8.1 Scalability Challenges

As LLM-based multi-agent systems transition from laboratory prototypes to production deployments, scalability emerges as a fundamental bottleneck that permeates architectural design, coordination mechanisms, and computational infrastructure.

8.1.1 Agent Population Scaling

Current LLM-MAS research predominantly focuses on systems with fewer than 10 agents, yet many real-world applications require coordination among hundreds or thousands of autonomous entities. Orogat et al. [2026] demonstrate that framework-level design choices alone can increase latency by over 100× when scaling from small to large agent populations, while coordination success rates can drop from above 90% to below 30%. This degradation stems from several interconnected factors:

Communication Complexity. The communication overhead in multi-agent systems typically scales quadratically ($O(n^2)$) with the number of agents in broadcast-based architectures. While targeted communication patterns reduce this to linear scaling, they introduce coordination challenges in determining optimal message routing. The K-Step Reachability Protocol proposed by Cheng et al. [2026] represents an initial attempt to address this by constraining message passing to physically accessible neighbors, but general solutions for arbitrary network topologies remain elusive.

Context Window Limitations. LLMs face inherent constraints on context length, which become acute in multi-agent settings where agents must maintain awareness of peer communications, shared memories, and environmental states. Current workarounds include hierarchical summarization and selective attention mechanisms, but these introduce information loss that compounds with system scale. The persistent memory architectures proposed by Shkolnikov [2026] and Mody et al. [2026] offer promising directions but remain limited in their capacity to handle large-scale deployments.

Coordination Overhead. As agent populations grow, the complexity of achieving consensus increases nonlinearly. Jo and Park [2025] demonstrate that Byzantine-robust coordination protocols can maintain system integrity with faulty agents, but their convergence time scales poorly with population size. Similarly, the CRDT-based coordination mechanisms explored by Pugachev [2025] provide lock-free concurrent operations but require careful conflict resolution strategies that become increasingly complex at scale.

8.1.2 Task Complexity Scaling

Beyond agent population, the complexity of tasks themselves presents scaling challenges. Long-horizon planning requires maintaining coherent goal representations over extended time periods, while multi-step reasoning demands reliable memory and reflection mechanisms. Huang et al. [2024b] identify five key planning dimensions—task decomposition, plan selection, external module integration, reflection, and memory—each of which presents unique scaling difficulties.

The SCALAR framework [Zabounidis et al., 2026] demonstrates that hierarchical skill composition can enable complex task execution, but the skill libraries required for diverse domains remain largely unexplored. Similarly, the plan reuse mechanisms proposed by Li et al. [2025a] achieve significant latency reductions but depend on accurate intent classification that becomes increasingly challenging with task diversity.

8.1.3 Promising Directions for Scalability

- **Hierarchical Federation:** Multi-level organizational structures that delegate coordination responsibilities to local subgroups, reducing global communication overhead while maintaining system coherence.
- **Sparse Communication Protocols:** Learned communication policies that minimize message exchange while preserving coordination quality, potentially leveraging graph neural architectures for adaptive routing.
- **Compositional Memory:** Distributed memory systems that decompose knowledge across specialized stores, enabling selective retrieval without requiring monolithic context maintenance.
- **Efficient Agent Architectures:** Specialized LLM variants optimized for multi-agent roles, potentially smaller models fine-tuned for specific coordination tasks rather than general-purpose reasoning.

8.2 Robustness Challenges

Robustness in LLM-based multi-agent systems encompasses resilience to environmental perturbations, agent failures, adversarial manipulation, and distributional shift. The interconnected nature of these systems creates failure modes that differ fundamentally from single-agent architectures.

8.2.1 Communication Failures and Information Asymmetry

Multi-agent systems rely critically on inter-agent communication, making them vulnerable to message loss, delay, and corruption. Peng et al. [2025] investigate collaboration under information asymmetry, demonstrating that even partial communication failures can significantly degrade task performance. The "echoing" phenomenon identified by Shekkizhar et al. [2025]—where agents progressively lose their individual identities through repeated interaction—represents a particularly insidious failure mode that emerges from seemingly benign communication patterns.

Byzantine Fault Tolerance. The presence of malicious or malfunctioning agents poses existential risks to multi-agent coordination. Jo and Park [2025] propose decentralized coordination protocols that maintain consensus despite Byzantine agents, but their approach assumes synchronous communication and known agent populations—assumptions that often fail in open-world deployments.

8.2.2 Adversarial Robustness

LLM-based agents inherit the vulnerabilities of their underlying language models while introducing new attack surfaces through their tool-use capabilities and inter-agent communication channels. Yang et al. [2024b] demonstrate that backdoor attacks can be embedded in agent training data, enabling adversaries to trigger malicious behaviors through specific input patterns. More concerningly, Chen et al. [2026] show that planning agents can be manipulated through carefully crafted user requests that exploit their helpfulness orientation.

The AgentGuard framework addresses unsafe tool orchestration through automated workflow discovery, but comprehensive adversarial robustness for multi-agent systems remains an open challenge. The attack surface expands with each additional agent, tool, and communication channel, creating combinatorial complexity in security analysis.

8.2.3 Environmental and Distributional Shift

Multi-agent systems deployed in dynamic environments must contend with non-stationary state distributions that arise from both environmental changes and the adaptive behaviors of peer agents. Cheng et al. [2026] address interference-aware communication in dynamic topologies, but general solutions for distributional shift in multi-agent contexts remain underdeveloped.

8.2.4 Promising Directions for Robustness

- **Redundant Coordination Pathways:** Multi-modal communication channels that provide fallback coordination mechanisms when primary pathways fail, inspired by biological systems' degeneracy principles.

- **Formal Verification Methods:** Techniques for proving safety and liveness properties of multi-agent protocols, potentially adapting model checking approaches to handle the stochastic nature of LLM-based reasoning.

- **Adversarial Training Pipelines:** Multi-agent adversarial training frameworks that expose systems to diverse attack strategies during development, building robustness through controlled vulnerability discovery.

- **Continuous Monitoring and Adaptation:** Runtime systems that detect distributional shift and trigger model adaptation or human intervention before performance degrades catastrophically.

8.3 Interpretability Challenges

Understanding the decision-making processes of LLM-based multi-agent systems presents unique challenges that extend beyond the interpretability concerns of individual models. The emergent behaviors arising from agent interactions, the distributed nature of reasoning across specialized roles, and the temporal dynamics of multi-turn coordination all demand new interpretability paradigms.

8.3.1 Emergent Behavior Interpretation

Erisken et al. [2025] demonstrate that multi-agent ensembles exhibit behaviors fundamentally unpredictable from isolated agent analysis. Their Multi-Agent Emergent Behavior Evaluation framework reveals phenomena such as peer pressure influencing convergence and emergent consensus-seeking behaviors that cannot be attributed to individual agent policies. This presents a critical interpretability gap: we cannot explain system behavior by aggregating explanations of individual agents.

Attribution Across Agents. When a multi-agent system produces an undesirable outcome, determining which agent(s) bear responsibility becomes non-trivial. The causal chain may span multiple agents, with each contributing partial decisions that collectively produce the failure. Current attribution methods focus on single-model explanations and lack the compositional reasoning necessary for multi-agent contexts.

8.3.2 Coordination Transparency

The communication protocols governing inter-agent interaction often operate as black boxes, with agents developing idiosyncratic message formats and implicit coordination strategies that resist human interpretation. Peters et al. [2025] survey the field of emergent language in multi-agent systems, highlighting the tension between communication efficiency and interpretability: optimized protocols often sacrifice human-readability for compactness.

8.3.3 Decision Process Traceability

Understanding why a multi-agent system made a particular decision requires tracing reasoning across multiple agents, communication rounds, and potential tool invocations. The Agent-as-a-Judge framework [Zhuge et al., 2024] provides mechanisms for agentic evaluation, but the evaluation process itself introduces additional interpretability challenges.

8.3.4 Promising Directions for Interpretability

- **Causal Attribution Networks:** Methods for tracing causal influence through multi-agent interaction graphs, potentially adapting techniques from causal inference and counterfactual reasoning.

- **Structured Communication Protocols:** Standardized message formats that enforce interpretability constraints while preserving expressiveness, building on the protocol standardization efforts surveyed by Ehtesham et al. [2025].

- **Multi-Agent Explanation Frameworks:** Explanation methods designed specifically for collective decision-making, potentially involving hierarchical summarization of agent reasoning processes.

- **Interactive Debugging Tools:** Development environments that allow practitioners to inspect and modify agent states, communication patterns, and decision processes at runtime.

8.4 Safety and Alignment Challenges

The deployment of autonomous multi-agent systems in high-stakes domains raises profound safety and alignment concerns that extend beyond those of individual AI systems. The interaction dynamics between agents can amplify risks, create novel failure modes, and complicate value alignment in ways that current frameworks cannot adequately address.

8.4.1 Multi-Agent Alignment

Traditional alignment approaches assume a single AI system interacting with human stakeholders. Multi-agent systems introduce the challenge of ensuring that all agents remain aligned with human values while potentially pursuing different sub-goals and operating with different information. Xu et al. [2025] analyze catastrophic risks in autonomous LLM agents, demonstrating that the trade-offs between helpfulness, harmlessness, and honesty (HHH) become increasingly complex in multi-agent settings.

Value Conflict Resolution. When agents with different specializations encounter situations where their objectives conflict, the resolution mechanism itself must be aligned with human values. The constitutional governance framework proposed by de Curtò and de Zarzà [2026] introduces ethical cooperation scores for multi-agent systems, but the specification of constitutional constraints remains a significant challenge.

Emergent Misalignment. Erisken et al. [2025] demonstrate that individually aligned agents can produce collectively misaligned outcomes through interaction dynamics. This emergent misalignment cannot be prevented by aligning agents in isolation—it requires alignment frameworks that account for multi-agent interaction.

8.4.2 Tool Use Safety

LLM-based agents increasingly interact with external tools and APIs, creating safety risks that scale with the capabilities granted to the system. Chen et al. [2026] identify “user-mediated attacks” where adversaries exploit agents’ helpfulness orientation to induce unsafe behaviors. The AgentGuard framework addresses this through autonomous discovery of unsafe workflows, but comprehensive tool-use safety remains an open problem.

8.4.3 Catastrophic Risk Analysis

The potential for multi-agent systems to cause severe harm—whether through coordinated action, cascading failures, or emergent behaviors—demands rigorous risk analysis frameworks. Xu et al. [2025] evaluate catastrophic risks including CBRN (chemical, biological, radiological, nuclear) scenarios, revealing that multi-agent systems can amplify both beneficial and harmful capabilities in ways that single-agent analysis cannot predict.

8.4.4 Promising Directions for Safety and Alignment

- **Multi-Agent Constitutional AI:** Extensions of constitutional AI methods to multi-agent settings, potentially involving shared constitutions, distributed oversight, and inter-agent accountability mechanisms.

- **Formal Safety Specifications:** Mathematical frameworks for specifying and verifying safety properties in multi-agent systems, potentially adapting formal methods from distributed systems and control theory.

- **Human-in-the-Loop Governance:** Architectural patterns that maintain meaningful human oversight over multi-agent systems without creating bottlenecks that negate the benefits of autonomy.

- **Adversarial Safety Testing:** Systematic methodologies for identifying and mitigating emergent risks in multi-agent systems, potentially involving red-team exercises and automated vulnerability discovery.

8.5 Evaluation Challenges

The evaluation of LLM-based multi-agent systems presents methodological challenges that fundamentally differ from single-agent evaluation. Traditional benchmarks assume isolated task completion, but multi-agent systems require assessment of coordination, communication, emergent behavior, and system-level properties.

8.5.1 Benchmark Limitations

Current multi-agent benchmarks suffer from several critical limitations. Mohammadi et al. [2025] propose a two-dimensional taxonomy for LLM agent evaluation (objectives $\times$ process), but note that existing benchmarks cover only narrow slices of this space. The MAFBench framework [Orogat et al., 2026] represents progress toward unified evaluation, but significant gaps remain:

Domain Coverage. Most benchmarks focus on specific domains (code generation, game playing, web navigation) with limited transferability to the diverse applications where multi-agent systems are deployed. Domain-specific evaluation suites like ZeroDayBench [Lau et al., 2026] for cybersecurity and GeoBenchX [Krechetova and Kochedykov, 2025] for geospatial tasks address particular domains but cannot capture cross-domain generalization.

Process Evaluation. Evaluating the process by which multi-agent systems achieve outcomes—not just the outcomes themselves—is essential for understanding system behavior. The AEMA framework [Lee et al., 2026] provides process-aware evaluation with auditable records, but process evaluation methodologies remain nascent.

8.5.2 Metric Development

Multi-agent systems require metrics that capture coordination quality, communication efficiency, consensus achievement, and emergent behavior—properties not captured by traditional accuracy or success rate metrics. Reza [2025] introduce psychometric evaluation frameworks for multi-agent debate, revealing that agents exhibit stable, measurable psychometric profiles that influence collective outcomes.

Coordination Metrics. Metrics for coordination quality must capture both efficiency (minimizing communication overhead) and effectiveness (achieving shared goals). The HumanEvalComm benchmark [Wu and Fard, 2025] evaluates communication competence in code generation, but general coordination metrics remain underdeveloped.

8.5.3 Emergent Behavior Evaluation

Perhaps the most significant evaluation challenge is assessing emergent behaviors—system properties that arise from agent interactions but cannot be predicted from individual agent capabilities. Erisken et al. [2025] propose frameworks for emergent behavior evaluation, but systematic methodologies remain elusive.

8.5.4 Promising Directions for Evaluation

- **Comprehensive Benchmark Suites:** Unified evaluation frameworks that systematically assess coordination, communication, planning, and emergent behavior across diverse domains and scales.

- **Process-Aware Metrics:** Evaluation methodologies that capture the trajectory of multi-agent reasoning, not just final outcomes, enabling diagnosis of failure modes and optimization opportunities.

- **Human-AI Collaborative Evaluation:** Frameworks that integrate human judgment into multi-agent evaluation, potentially leveraging the Agent-as-a-Judge paradigm while maintaining human oversight.

- **Continuous Evaluation Pipelines:** Infrastructure for ongoing evaluation of deployed multi-agent systems, detecting performance drift and emerging failure modes in production environments.

8.6 Theoretical Foundations

Despite empirical advances in LLM-based multi-agent systems, theoretical understanding lags significantly behind practical capabilities. Foundational questions about convergence, optimality, and emergent properties remain largely open.

8.6.1 Convergence and Stability Analysis

Multi-agent systems exhibit complex dynamics that resist analysis through traditional control theory or game theory frameworks. The non-stationarity introduced by learning agents, the stochastic nature of LLM reasoning, and the discrete structure of language-based communication all complicate theoretical analysis.

Consensus Convergence. Under what conditions do multi-agent systems converge to consensus? Jo and Park [2025] provide convergence guarantees for Byzantine-robust protocols under specific assumptions, but general conditions for convergence in LLM-based systems remain unknown.

Stability Guarantees. Can we provide formal guarantees that multi-agent systems will remain within safe operating regions? Shielded Multi-Agent Reinforcement Learning frameworks demonstrate that probabilistic logic shields can enforce safety constraints [Ahmed et al., 2022], but the computational complexity of verification scales poorly with system size.

8.6.2 Game-Theoretic Foundations

LLM-based agents engage in strategic interactions that can be analyzed through game-theoretic lenses, but standard game theory assumes fixed strategy spaces and utility functions—assumptions violated by the open-ended reasoning capabilities of LLMs. Park et al. [2024] analyze no-regret learning in LLM agents, revealing both alignment with and deviation from theoretical predictions.

Equilibrium Analysis. What equilibrium concepts are appropriate for LLM-based multi-agent systems? Standard Nash equilibrium assumes common knowledge of rationality, but LLM agents may exhibit bounded rationality, irrational preferences, or capability heterogeneity that invalidates standard equilibrium analysis.

8.6.3 Emergence Theory

The emergence of collective behaviors from individual agent interactions represents perhaps the most profound theoretical gap. Peters et al. [2025] survey emergent language in multi-agent systems, but a general theory of emergence in LLM-based systems—predicting what collective behaviors will arise from given agent architectures and interaction protocols—remains absent.

8.6.4 Complexity Analysis

The computational complexity of multi-agent coordination problems is well-understood for classical settings, but LLM-based systems introduce novel complexity considerations. The cost of LLM inference, the structure of natural language communication, and the role of prompting strategies all influence tractability in ways not captured by traditional complexity analysis.

8.6.5 Promising Directions for Theoretical Foundations

- **Probabilistic Multi-Agent Reasoning:** Theoretical frameworks that account for the stochastic, approximate nature of LLM reasoning, potentially adapting probabilistic verification methods from formal methods.
- **Bounded Rationality Models:** Game-theoretic models that incorporate the bounded rationality of LLM agents, potentially drawing on behavioral game theory and cognitive science.
- **Emergence Taxonomies:** Systematic classification of emergent phenomena in multi-agent systems, enabling prediction of when specific collective behaviors will arise.
- **Complexity-Theoretic Analysis:** Complexity analysis specifically designed for LLM-based multi-agent systems, accounting for inference costs, communication structure, and prompt engineering.

8.7 Cross-Cutting Research Directions

Beyond domain-specific challenges, several cross-cutting research directions have the potential to advance the field substantially.

8.7.1 Standardization Efforts

The proliferation of incompatible frameworks, protocols, and evaluation methodologies fragments research efforts and impedes reproducibility. The protocol standardization efforts surveyed by Ehtesham et al. [2025]—including MCP, ACP, A2A, and ANP—represent progress toward interoperability, but broader standardization of architectures, benchmarks, and metrics is needed.

8.7.2 Human-Agent Collaboration

Most multi-agent research focuses on fully autonomous systems, but many practical applications involve human-agent teams. Zou et al. [2025] survey human-agent collaboration, highlighting the need for interaction patterns that leverage complementary human and AI capabilities. Comprehensive frameworks for human-agent teaming that maintain meaningful human oversight while preserving the benefits of agent autonomy remain underdeveloped.

8.7.3 Domain-Specific Architectures

While general-purpose multi-agent frameworks provide flexibility, domain-specific architectures may offer superior performance for particular applications. The success of TradingAgents [Xiao et al., 2024] and FinCon [Yu et al., 2024] in financial domains, MedAide [Yang et al., 2024a] in healthcare, and PhishDebate [Li et al., 2025c] in cybersecurity suggests that domain expertise can inform architectural design.

8.7.4 Open Research Infrastructure

The field would benefit from shared research infrastructure including:

- Standardized agent interfaces and communication protocols
- Comprehensive benchmark suites with leaderboards
- Open-source reference implementations of major frameworks
- Shared evaluation infrastructure for reproducible experiments

8.8 Conclusion

The challenges facing LLM-based multi-agent systems are substantial but not insurmountable. Progress will require coordinated efforts across multiple research communities—natural language processing, multi-agent systems, reinforcement learning, formal methods, and human-computer interaction. The rapid pace of empirical advances provides both motivation and raw material for theoretical progress, while the urgent practical applications driving this research ensure that solutions will have immediate impact.

The research directions outlined in this section—scalability through hierarchical federation, robustness through redundant coordination, interpretability through causal attribution, safety through constitutional governance, evaluation through comprehensive benchmarks, and theoretical foundations through probabilistic reasoning—represent promising paths forward. However, the interconnected nature of these challenges suggests that progress will require holistic approaches that address multiple dimensions simultaneously.

As LLM-based multi-agent systems transition from research curiosities to deployed infrastructure, addressing these challenges becomes not merely an academic exercise but a societal imperative. The potential benefits—accelerated scientific discovery, enhanced human productivity, and novel problem-solving capabilities—are matched by potential risks—misaligned objectives, emergent failures, and concentration of autonomous capability. Navigating this landscape requires the rigorous, collaborative, and forward-looking research program that this survey aims to inspire.

9 Conclusion

This survey has presented a comprehensive examination of LLM-based multi-agent systems (LLM-MAS), a rapidly emerging paradigm that represents a fundamental shift in how we conceptualize, design, and deploy autonomous AI systems. Through our systematic analysis of over 100 research papers spanning 2022–2026, we have traced the evolution of this field from early architectural explorations to sophisticated frameworks capable of addressing complex real-world challenges. In this concluding section, we synthesize our key findings, highlight the contributions of this survey, and offer perspectives on the trajectory of LLM-MAS research.

9.1 Key Findings

Our analysis reveals several fundamental insights that characterize the current state and future potential of LLM-based multi-agent systems.

Architectural Maturation. The field has witnessed a rapid maturation of architectural paradigms, transitioning from simple peer-to-peer agent configurations to sophisticated hierarchical and self-organizing structures. The modular decomposition pattern—exemplified by planner-caller-summarizer architectures [Shen et al., 2024]—has emerged as a particularly effective approach for leveraging specialized capabilities. Meanwhile, tree-of-agents structures [Chen et al., 2024] and hierarchical auto-organizing systems [Zhao et al., 2024] demonstrate the potential for scalable, adaptive multi-agent deployments that can respond dynamically to task requirements.

Coordination as a Critical Challenge. Coordination mechanisms remain among the most actively researched areas in LLM-MAS. Our taxonomy identifies communication protocols, coordination strategies, and consensus mechanisms as three interconnected dimensions that collectively determine system effectiveness. The emergence of standardized protocols—including MCP, ACP, A2A, and ANP [Ehtesham et al., 2025]—signals a growing recognition that interoperability and standardized communication are prerequisites for widespread adoption. However, challenges persist in Byzantine-robust coordination [Jo and Park, 2025], information asymmetry handling [Peng et al., 2025], and the prevention of emergent failures such as the "echoing" phenomenon [Shekkizhar et al., 2025].

Planning Integration. The integration of LLM-based reasoning with classical planning formalisms represents a significant research direction. Our review identifies task decomposition, plan selection, external module integration, reflection, and memory as the five pillars of LLM agent planning [Huang et al., 2024b]. The development of frameworks such as AgentGen [Hu et al., 2024], NL2Plan [Gestrin et al., 2024], and SCALAR [Zabounidis et al., 2026] demonstrates productive synergies between LLM capabilities and structured planning approaches. These hybrid systems offer the flexibility of natural language understanding combined with the rigor of formal planning methods.

Evaluation Complexity. The evaluation of LLM-MAS presents methodological challenges that fundamentally differ from single-agent assessment. Traditional benchmarks assume isolated task completion, but multi-agent systems require assessment of coordination quality, communication efficiency, consensus achievement, and emergent behavior. Frameworks such as MAF-Bench [Orogat et al., 2026], Agent-as-a-Judge [Zhuge et al., 2024], and AEMA [Lee et al., 2026] represent important steps toward comprehensive evaluation, yet significant gaps remain in benchmark coverage, process evaluation, and emergent behavior assessment.

Application-Driven Innovation. Perhaps the most compelling evidence for the maturity of LLM-MAS is the breadth and depth of domain-specific applications. From financial trading [Xiao et al., 2024, Yu et al., 2024] to healthcare [Yang et al., 2024a], from scientific discovery [Rothfarb et al., 2025, He et al., 2025] to cybersecurity [Li et al., 2025c, Lau et al., 2026], multi-agent systems are demonstrating tangible value across diverse application domains. These implementations reveal that domain expertise can inform architectural design, leading to specialized frameworks that outperform general-purpose alternatives.

9.2 Contributions of This Survey

This survey makes several contributions to the growing body of LLM-MAS literature:

Comprehensive Taxonomy. We have developed a multi-dimensional taxonomy that classifies LLM-MAS research across architecture paradigms, coordination mechanisms, planning strategies, evaluation metrics, and application domains. This taxonomy provides researchers and practitioners with a systematic framework for understanding the design space and identifying relevant prior work.

Unified Perspective. By synthesizing insights from natural language processing, multi-agent systems, reinforcement learning, and human-computer interaction, we offer a unified perspective that transcends disciplinary boundaries. This cross-cutting analysis reveals connections and opportunities that might not be apparent when examining individual research threads in isolation.

Research Roadmap. Our identification of open challenges—scalability, robustness, interpretability, safety and alignment, evaluation, and theoretical foundations—provides a research roadmap for the field. These challenges are not merely obstacles but opportunities for significant contributions that will shape the future of autonomous systems.

Practical Guidance. For practitioners seeking to deploy LLM-MAS in real-world applications, this survey offers practical guidance on architectural selection, coordination mechanism design, and evaluation methodology. The classification guidelines and recommended reading orders provide actionable resources for both newcomers and experienced researchers.

9.3 Significance of LLM-MAS Research

The significance of LLM-based multi-agent systems extends beyond technical achievements to encompass broader implications for AI research and society.

Paradigm Shift in AI Architecture. LLM-MAS represents a paradigm shift from monolithic AI systems to distributed, modular architectures. This transition mirrors broader trends in software engineering toward microservices and distributed systems, suggesting that the future of AI may lie not in ever-larger single models but in carefully orchestrated ensembles of specialized agents. The implications for scalability, maintainability, and adaptability are profound.

Bridging Symbolic and Neural Approaches. Multi-agent systems offer a natural framework for integrating symbolic reasoning with neural capabilities. Agents can serve as bridges between different representational paradigms, with some agents handling structured knowledge while others process unstructured inputs. This synthesis addresses a long-standing challenge in AI and may prove essential for achieving robust, interpretable intelligence.

Human-AI Collaboration. The multi-agent paradigm provides natural interfaces for human-AI collaboration. Rather than treating humans as external operators, future systems may include human agents as equal participants in multi-agent ensembles. This perspective aligns with emerging research on human-agent teaming [Zou et al., 2025] and suggests pathways toward AI systems that complement rather than replace human capabilities.

Societal Impact. As LLM-MAS deployments become more prevalent, their societal impact will grow correspondingly. The potential for accelerated scientific discovery, enhanced productivity, and novel problem-solving capabilities must be weighed against risks of misaligned objectives, emergent failures, and concentration of autonomous capability. Responsible development requires proactive engagement with these considerations.

9.4 The Field's Trajectory

Looking forward, we identify several trends that will likely shape the trajectory of LLM-MAS research:

From Frameworks to Standards. The proliferation of frameworks, protocols, and evaluation methodologies will likely give way to consolidation around established standards. The protocol standardization efforts surveyed by Ehtesham et al. [2025] represent early steps in this direction. Standardized interfaces will enable interoperability, facilitate reproducibility, and accelerate progress by allowing researchers to build on common foundations.

From Demonstration to Deployment. Current research focuses heavily on proof-of-concept demonstrations in controlled environments. The coming years will see increased emphasis on deployment in production settings, with corresponding attention to reliability, monitoring, and maintenance. This transition will demand advances in robustness, safety, and operational practices that go beyond current research priorities.

From Empirical to Theoretical. The rapid empirical progress in LLM-MAS has outpaced theoretical understanding. We anticipate growing interest in foundational questions: Under what conditions do multi-agent systems converge? What equilibrium concepts are appropriate for LLM-based agents? How can we predict and control emergent behaviors? Answering these questions will require new theoretical frameworks that account for the unique properties of LLM-based reasoning.

From Autonomous to Collaborative. While current research focuses primarily on fully autonomous systems, the future lies in human-agent collaboration. Systems that can effectively integrate human expertise, maintain meaningful human oversight, and adapt to human preferences will be essential for high-stakes applications. This shift demands new interaction paradigms, trust mechanisms, and governance structures.

From General to Specialized. The trend toward domain-specific architectures—evident in financial, healthcare, and scientific applications—will continue. Specialized agents, trained or prompted for particular domains, will likely outperform general-purpose alternatives in their areas of expertise. This specialization raises important questions about the boundaries of agent capabilities and the design of effective multi-domain systems.

9.5 Closing Remarks

LLM-based multi-agent systems stand at an inflection point. The foundational capabilities are established, the application potential is demonstrated, and the research community is rapidly expanding. Yet the most significant challenges—scalability to hundreds of agents, robust coordination under adversarial conditions, formal safety guarantees, and theoretical understanding of emergence—remain open.

The promise of LLM-MAS lies not merely in aggregating the capabilities of individual agents but in the emergent possibilities that arise from their interaction. Just as biological systems achieve complexity through the coordination of simpler components, and just as human organizations accomplish tasks beyond individual capacity through collaboration, LLM-MAS offers a pathway to AI capabilities that transcend the limitations of any single model.

Realizing this promise requires sustained effort across multiple dimensions: technical advances in architecture and coordination, methodological advances in evaluation and benchmarking, theoretical advances in understanding emergence and convergence, and practical advances in deployment and governance. The challenges are substantial, but so too are the potential benefits—accelerated scientific discovery, enhanced human productivity, and AI systems that are more robust, interpretable, and aligned with human values.

This survey has aimed to provide a foundation for these efforts by mapping the current landscape, identifying key challenges, and charting promising directions. As the field continues its rapid evolution, we hope this work serves as both a reference point for understanding what has been achieved and a guide for what remains to be accomplished. The journey toward truly capable, safe, and beneficial multi-agent AI systems has only begun; the most exciting developments lie ahead.

"The whole is greater than the sum of its parts."
— Aristotle, *Metaphysica*

Acknowledgments

This work was supported in part by [anonymous funding sources]. We thank the anonymous reviewers for their valuable feedback.

References

Mohamed Aghzal, Gregory J. Stein, and Ziyu Yao. Why do llm-based web agents fail? a hierarchical planning perspective. *arXiv preprint arXiv:2603.14248*, 2026.

Ibrahim H. Ahmed, Cillian Brewitt, Ignacio Carlucho, Filippos Christianos, Mhairi Dunion, et al. Deep reinforcement learning for multi-agent interaction. *AI Communications Special Issue on Multi-Agent Systems Research in the UK*, 2022.

Anurag Ajay, Seungwook Han, Yilun Du, Shuang Li, Abhi Gupta, et al. Compositional foundation models for hierarchical planning. *arXiv preprint arXiv:2309.08587*, 2023.

P. Berry, T. J. Lee, and D. E. Wilkins. Interactive execution monitoring of agent teams. *Journal of Artificial Intelligence Research*, 18:217–261, 2003.

Gaowei Chang, Eidan Lin, Chengxuan Yuan, Rizhao Cai, Binbin Chen, et al. Agent network protocol technical white paper. *arXiv preprint arXiv:2508.00007*, 2025.

Fengchao Chen, Tingmin Wu, Van Nguyen, and Carsten Rudolph. Too helpful to be safe: User-mediated attacks on planning and web-use agents. *arXiv preprint arXiv:2601.10758*, 2026.

Jiaqi Chen, Yuxian Jiang, Jiachen Lu, and Li Zhang. S-agents: Self-organizing agents in open-ended environments. *arXiv preprint arXiv:2402.04578*, 2024.

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chen Liu, et al. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors. *arXiv preprint arXiv:2308.10848*, 2023.

Ziyu Cheng, Jinsheng Ren, Zhouxian Jiang, Chenzhihang Li, Rongye Shi, et al. Interference-aware k-step reachable communication in multi-agent reinforcement learning. *arXiv preprint arXiv:2603.15054*, 2026.

Kun Chu, Xufeng Zhao, Cornelius Weber, and Stefan Wermter. Llm+map: Bimanual robot task planning using large language models and planning domain definition language. *arXiv preprint arXiv:2503.17309*, 2025.

J. de Curtò and I. de Zarzà. Llm constitutional multi-agent governance. *arXiv preprint arXiv:2603.13189*, 2026.

Abul Ehtesham, Aditi Singh, Gaurav Kumar Gupta, and Saket Kumar. A survey of agent interoperability protocols: Model context protocol (mcp), agent communication protocol (acp), agent-to-agent protocol (a2a), and agent network protocol (anp). *arXiv preprint arXiv:2505.02279*, 2025.

Sinem Erisken, Timothy Gothard, Martin Leitgab, and Ram Potham. Maebe: Multi-agent emergent behavior framework. *arXiv preprint arXiv:2506.03053*, 2025.

FIPA. Fipa acl message structure specification. *Foundation for Intelligent Physical Agents*, 2002. Document SC00061G.

Saman Forouzandeh, Wei Peng, Parham Moradi, Xinghuo Yu, and Mahdi Jalili. Learning hierarchical procedural memory for llm agents through bayesian selection and contrastive refinement. *arXiv preprint arXiv:2512.18950*, 2025.

Elliot Gestrin, Marco Kuhlmann, and Jendrik Seipp. Nl2plan: Robust llm-driven planning from minimal text descriptions. *arXiv preprint arXiv:2405.04215*, 2024.

Yiduo Guo, Zekai Zhang, Yaobo Liang, Dongyan Zhao, and Nan Duan. Pptc benchmark: Evaluating large language models for powerpoint task completion. *arXiv preprint arXiv:2311.01767*, 2023.

Muhammad Haseeb. Context engineering for multi-agent llm code assistants using elicit, notebooklm, chatgpt, and claude code. *arXiv preprint arXiv:2508.08322*, 2025.

Xin He, Liangliang You, Hongduan Tian, Bo Han, Ivor Tsang, et al. Lang-pinn: From language to physics-informed neural networks via a multi-agent framework. *arXiv preprint arXiv:2510.05158*, 2025.

Mengkang Hu, Pu Zhao, Can Xu, Qingfeng Sun, Jianguang Lou, et al. Agentgen: Enhancing planning abilities for large language model based agent via environment and task generation. *arXiv preprint arXiv:2408.00764*, 2024.

Sukai Huang, Trevor Cohn, and Nir Lipovetzky. Chasing progress, not perfection: Revisiting strategies for end-to-end llm plan generation. *arXiv preprint arXiv:2412.10675*, 2024a.

Xu Huang, Weiwen Liu, Xiaolong Chen, Xingmei Wang, Hao Wang, et al. Understanding the planning of llm agents: A survey. *arXiv preprint arXiv:2402.02716*, 2024b.

Nhat-Minh Huynh, Hoang-Giang Cao, and I-Chen Wu. Multi-agent training for pommerman: Curriculum learning and population-based self-play approach. *arXiv preprint arXiv:2407.00662*, 2024.

Yongrae Jo and Chanik Park. Byzantine-robust decentralized coordination of llm agents. *arXiv preprint arXiv:2507.14928*, 2025.

Varvara Krechetova and Denis Kochedykov. Geobenchx: Benchmarking llms in agent solving multistep geospatial tasks. *arXiv preprint arXiv:2503.18129*, 2025.

Yasser Labrou and Tim Finin. Semantics for an agent communication language. *Proceedings of the Fourth International Workshop on Agent Theories, Architectures, and Languages (ATAL)*, pages 209–214, 1997.

Nancy Lau, Louis Sloot, Jyoutir Raj, Giuseppe Marco Boscardin, Evan Harris, et al. Zerodaybench: Evaluating llm agents on unseen zero-day vulnerabilities for cyberdefense. *arXiv preprint arXiv:2603.02297*, 2026.

YenTing Lee, Keerthi Koneru, Zahra Moslemi, Sheethal Kumar, and Ramesh Radhakrishnan. Aema: Verifiable evaluation framework for trustworthy and controlled agentic llm systems. *arXiv preprint arXiv:2601.11903*, 2026.

Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. Camel: Communicative agents for "mind" exploration of large language model society. *arXiv preprint arXiv:2303.17760*, 2023.

Guopeng Li, Ruiqi Wu, and Haisheng Tan. A plan reuse mechanism for llm-driven agent. *arXiv preprint arXiv:2512.21309*, 2025a.

Ning Li, Xiangmou Qu, Jiamu Zhou, Jun Wang, Muning Wen, et al. Mobileuse: A gui agent with hierarchical reflection for autonomous mobile operation. *arXiv preprint arXiv:2507.16853*, 2025b.

Wenhao Li, Selvakumar Manickam, Yung-wey Chong, and Shankar Karuppayah. Phishdebate: An llm-based multi-agent framework for phishing website detection. *arXiv preprint arXiv:2506.15656*, 2025c.

Xin Li, Mengbing Liu, and Chau Yuen. Llm agent communication protocol (lacp) requires urgent standardization: A telecom-inspired protocol is necessary. *arXiv preprint arXiv:2510.13821*, 2025d.

Yin Li. Decomposing llm self-correction: The accuracy-correction paradox and error depth hypothesis. *arXiv preprint arXiv:2601.00828*, 2025.

Jonathan Light, Min Cai, Sheng Shen, and Ziniu Hu. Avalonbench: Evaluating llms playing the game of avalon. *arXiv preprint arXiv:2310.05036*, 2023.

Marianne Menglin Liu, Daniel Garcia, Fjona Parllaku, Vikas Upadhyay, Syed Fahad Allam Shah, et al. Toolscope: Enhancing llm agent tool use through tool merging and context-aware filtering. In *arXiv preprint arXiv:2510.20036*, 2025.

Xinghua Lou, Miguel Lázaro-Gredilla, Antoine Dedieu, Carter Wendelken, Wolfgang Lehrach, et al. Autoharness: Improving llm agents by automatically synthesizing a code harness. *arXiv preprint arXiv:2603.03329*, 2026.

Chih-Yao Ma, Jiasen Lu, Zuxuan Wu, Ghassan AlRegib, Zsolt Kira, et al. Self-monitoring navigation agent via auxiliary progress estimation. *arXiv preprint arXiv:1901.03035*, 2019.

Shaoguang Mao, Yuzhe Cai, Yan Xia, Wenshan Wu, Xun Wang, et al. Alympics: Llm agents meet game theory – exploring strategic decision-making with ai agents. *arXiv preprint arXiv:2311.03220*, 2023.

Pearl Mody, Mihir Panchal, Rishit Kar, Kiran Bhowmick, and Ruhina Karani. Cranimem: Cranial inspired gated and bounded memory for agentic systems. *ICLR 2026 Workshop on Memory for LLM-Based Agentic Systems*, 2026.

Mahmoud Mohammadi, Yipeng Li, Jane Lo, and Wendy Yip. Evaluation and benchmarking of llm agents: A survey. *arXiv preprint arXiv:2507.21504*, 2025.

Abdelghny Orogat, Ana Rostam, and Essam Mansour. Understanding multi-agent llm frameworks: A unified benchmark and experimental analysis. *arXiv preprint arXiv:2602.03128*, 2026.

Chanwoo Park, Xiangyu Liu, Asuman Ozdaglar, and Kaiqing Zhang. Do llm agents have regret? a case study in online learning and games. *arXiv preprint arXiv:2403.16843*, 2024.

Damien Pellier, Alexandre Albore, Humbert Fiorino, and Rafael Bailon-Ruiz. Hddl 2.1: Towards defining a formalism and a semantics for temporal htn planning. *International Workshop of Hierarchical Planning (ICAPS)*, 2023.

Run Peng, Ziqiao Ma, Amy Pang, Sikai Li, Zhang Xi-Jia, et al. Communication and verification in llm agents towards collaboration under information asymmetry. *arXiv preprint arXiv:2510.25595*, 2025.

Jannik Peters, Constantin Waubert de Puiseau, Hasan Tercan, Arya Gopikrishnan, Gustavo Adolpho Lucas De Carvalho, et al. Emergent language: A survey and taxonomy. *Autonomous Agents and Multi-Agent Systems*, 39:18, 2025.

Israel Puerta-Merino, Carlos Núñez-Molina, Pablo Mesejo, and Juan Fernández-Olivares. A roadmap to guide the integration of llms in hierarchical planning. *arXiv preprint arXiv:2501.08068*, 2025.

Sergey Pugachev. Codecrdt: Observation-driven coordination for multi-agent llm code generation. *arXiv preprint arXiv:2510.18893*, 2025.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yihong Dang, et al. Communicative agents for software development. *arXiv preprint arXiv:2307.07924*, 2024.

Zarreen Reza. The social laboratory: A psychometric framework for multi-agent llm evaluation. *arXiv preprint arXiv:2510.01295*, 2025.

Samuel Rothfarb, Megan C. Davis, Ivana Matanovic, Baikun Li, Edward F. Holby, et al. Hierarchical multi-agent large language model reasoning for autonomous functional materials discovery. *arXiv preprint arXiv:2512.13930*, 2025.

Russell Schwartz and Pratap Tokekar. Robust multi-agent task assignment in failure-prone and adversarial environments. *arXiv preprint arXiv:2007.00100*, 2020.

Shinas Shaji, Fabian Huppertz, Alex Mitrevski, and Sebastian Houben. From language to action: Can llm-based agents be used for embodied robot cognition? *arXiv preprint arXiv:2603.03148*, 2026.

Sarath Shekkizhar, Romain Cosentino, Adam Earle, and Silvio Savarese. Echoing: Identity failures when llm agents talk to each other. *arXiv preprint arXiv:2511.09710*, 2025.

Weizhou Shen, Chenliang Li, Hongzhan Chen, Ming Yan, Xiaojun Quan, et al. Small llms are weak tool learners: A multi-llm agent. *arXiv preprint arXiv:2401.07324*, 2024.

Yakov Pyotr Shkolnikov. Agent memory below the prompt: Persistent q4 kv cache for multi-agent llm inference on edge devices. In *arXiv preprint arXiv:2603.04428*, 2026.

Ekhlas Sonu, Yingke Chen, and Prashant Doshi. Individual planning in agent populations: Exploiting anonymity and frame-action hypergraphs. *arXiv preprint arXiv:1503.07220*, 2015.

Michael Theologitis and Dan Suciu. Thucy: An llm-based multi-agent system for claim verification across relational databases. *arXiv preprint arXiv:2512.03278*, 2025.

Junyang Wang, Haiyang Xu, Jiabo Ye, Ming Yan, Weizhou Shen, et al. Mobile-agent: Autonomous multi-modal mobile device agent with visual perception. *arXiv preprint arXiv:2401.16158*, 2024.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, et al. Chain-of-thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022.

Christoph Willibald and Dongheui Lee. Hierarchical task decomposition for execution monitoring and error recovery: Understanding the rationale behind task demonstrations. *International Journal of Robotics Research (IJRR)*, 2025.

Duo Wu, Jinghe Wang, Yuan Meng, Yanning Zhang, Le Sun, et al. Catp-llm: Empowering large language models for cost-aware tool planning. *arXiv preprint arXiv:2411.16313*, 2024.

Jie JW Wu and Fatemeh H Fard. Humanevalcomm: Benchmarking the communication competence of code generation for llms and llm agent. *ACM Transactions on Software Engineering and Methodology*, 2025.

Yijia Xiao, Edward Sun, Di Luo, and Wei Wang. Tradingagents: Multi-agents llm financial trading framework. *arXiv preprint arXiv:2412.20138*, 2024.

Quanting Xie, So Yeon Min, Pengliang Ji, Yue Yang, Tianyi Zhang, et al. Embodied-rag: General non-parametric embodied memory for retrieval and generation. *arXiv preprint arXiv:2409.18313*, 2024.

Rongwu Xu, Xiaojian Li, Shuo Chen, and Wei Xu. Nuclear deployed: Analyzing catastrophic risks in decision-making of autonomous llm agents. *arXiv preprint arXiv:2502.11355*, 2025.

Dingkang Yang, Jinjie Wei, Mingcheng Li, Jiyao Liu, Lihao Liu, et al. Medaide: Information fusion and anatomy of medical intents via llm-based agent collaboration. *arXiv preprint arXiv:2410.12532*, 2024a.

Wenkai Yang, Xiaohan Bi, Yankai Lin, Sishuo Chen, Jie Zhou, et al. Watch out for your agents! investigating backdoor threats to llm-based agents. In *NeurIPS 2024*, 2024b.

Shunyu Yao, Dianhao Yu, Jeffrey Zhao, Izhak Shafran, Thomas L Griffiths, et al. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Ye Ye. Task memory engine (tme): Enhancing state awareness for multi-step llm agent tasks. *arXiv preprint arXiv:2504.08525*, 2025.

Yangyang Yu, Zhiyuan Yao, Haohang Li, Zhiyang Deng, Yupeng Cao, et al. Fincon: A synthesized llm multi-agent system with conceptual verbal reinforcement for enhanced financial decision making. *arXiv preprint arXiv:2407.06567*, 2024.

Renos Zabounidis, Yue Wu, Simon Stepputtis, Woojun Kim, Yuanzhi Li, et al. Scalar: Learning and composing skills through llm guided symbolic planning and deep rl grounding. *arXiv preprint arXiv:2603.09036*, 2026.

Ziyao Zeng, Chen Liu, Tianyu Liu, Hao Wang, Xiatao Sun, et al. Rulesmith: Multi-agent llms for automated game balancing. *arXiv preprint arXiv:2602.06232*, 2026.

Fan Zhang, Shulin Tian, Ziqi Huang, Yu Qiao, and Ziwei Liu. Evaluation agent: Efficient and promptable evaluation framework for visual generative models. *arXiv preprint arXiv:2412.09645*, 2024a.

Zekai Zhang, Yiduo Guo, Yaobo Liang, Dongyan Zhao, and Nan Duan. Pptc-r benchmark: Towards evaluating the robustness of large language models for powerpoint task completion. *arXiv preprint arXiv:2403.03788*, 2024b.

Zhonghan Zhao, Kewei Chen, Dongxu Guo, Wenhao Chai, Tian Ye, et al. Hierarchical auto-organizing system for open-ended multi-agent navigation. *arXiv preprint arXiv:2403.08282*, 2024.

Wangchunshu Zhou, Yuchen Eleanor Jiang, Long Li, Jialong Wu, Tiannan Wang, et al. Agents: An open-source framework for autonomous language agents. *arXiv preprint arXiv:2309.07870*, 2023.

Changxi Zhu, Mehdi Dastani, and Shihan Wang. A survey of multi-agent deep reinforcement learning with communication. *Autonomous Agents and Multi-Agent Systems*, 38(1):4, 2024.

Mingchen Zhuge, Changsheng Zhao, Dylan Ashley, Wenyi Wang, Dmitrii Khizbullin, et al. Agent-as-a-judge: Evaluate agents with agents. *arXiv preprint arXiv:2410.10934*, 2024.

Henry Peng Zou, Wei-Chieh Huang, Yaozu Wu, Yankai Chen, Chunyu Miao, et al. Llm-based human-agent collaboration and interaction systems: A survey. *arXiv preprint arXiv:2505.00753*, 2025.

Table 10: Extended Benchmark Comparison

Benchmark	Domain	Tasks	Year
MAFBench	General	Multiple	2026
DevAI	Software Dev	55	2024
AvalonBench	Social Deduction	15+	2023
HumanEvalComm	Code Gen	164+	2024
ZeroDayBench	Security	100+	2026
PPTC/PPTC-R	Office Tasks	28	2023/24
GeoBenchX	Geospatial	50+	2025
CORE-Bench	Reproducibility	270	2024

A Detailed Benchmark Comparison

This appendix provides extended comparison tables for evaluation benchmarks discussed in Section 6.

B Protocol Specification Summary

This appendix summarizes the key specifications of emerging agent communication protocols.

Table 11: Agent Communication Protocol Specifications

Protocol	Layer Architecture	Decentralization	Primary Use Case
MCP	Application	Partial	Tool integration
ACP	Transport	Partial	Enterprise messaging
A2A	Network	Partial	Cross-platform discovery
ANP	Three-layer	High	Open agent networks
LACP	Telecom-inspired	Medium	Standardization

C Architecture Pattern Catalog

This appendix provides a catalog of architectural patterns identified in our analysis, with guidance on selection criteria.

Table 12: Architecture Pattern Selection Guide

Pattern	Best For	Scale	Complexity
Flat/P2P	Debate, consensus	High	Low
Manager-Worker	Structured workflows	Medium	Medium
Tree-of-Agents	Hierarchical tasks	Medium-High	Medium
Modular Pipeline	Sequential processing	Medium	Low
Self-Organizing	Dynamic environments	High	High

D Glossary of Terms

ACL	Agent Communication Language

ANP Agent Network Protocol

CRDT Conflict-free Replicated Data Type

HTN Hierarchical Task Network

LLM Large Language Model

LLM-MAS LLM-based Multi-Agent System

MCP Model Context Protocol

PDDL Planning Domain Definition Language

ReAct Reasoning and Acting

ToT Tree-of-Thought